2

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

ONE AND TWO DIMENSIONAL DISCRETE WAVELET TRANSFORMS

by

Joey E. Legaspi

September, 1992

Thesis Advisor: Professor Alex W. Lam

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b. OFFICE SYMBOL *(If applicable)* 39	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943-5000		7b. ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS *(City, State, and ZIP Code)*		10. SOURCE OF FUNDING NUMBERS

Program Element No.	Project No.	Task No.	Work Unit Accession Number

11. TITLE *(Include Security Classification)*
ONE AND TWO DIMENSIONAL DISCRETE WAVELT TRANSFORMS

12. PERSONAL AUTHOR(S) Joey E. Legaspi

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(year, month, day)*	15. PAGE COUNT
Master's Thesis	From To	September 1992	128

16. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17. COSATI CODES

FIELD	GROUP	SUBGROUP

18. SUBJECT TERMS *(continue on reverse if necessary and identify by block number)*

19. ABSTRACT *(continue on reverse if necessary and identify by block number)*

Fourier transform techniques have been the favored methods in the analysis of signal and systems. One major drawback of Fourier methods is the difficulty in analyzing transient and/or non-stationary behavior. Recent advances in the field of wavelet theory show much promise in alleviating these problems. This thesis considers the realizations of the wavelet decomposition and reconstruction algorithms for the discrete case. We also present a multiple-phase development as a second and possibly a preferable method for decomposing signals.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS REPORT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Professor Alex W. Lam	22b. TELEPHONE *(Include Area code)* (408) 646-3044	22c. OFFICE SYMBOL EC/LA

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

One and Two Dimensional Discrete Wavelet Transforms

by

Joey E. Legaspi
Lieutenant, United States Navy
B.S.E.E., United States Naval Academy, 1985

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 1992

Author: Joey E. Legaspi
Joey E. Legaspi

Approved by: [illegible]
Professor Alex W. Lam, Thesis Advisor

[illegible]
Professor Herschel H. Loomis, Jr., Second Reader

M. A. Morgan
Professor Michael A. Morgan, Chairman
Department of Electrical and Computer Engineering

ABSTRACT

Fourier transform techniques have been the favored methods in the analysis of signals and systems. One major drawback of Fourier methods is the difficulty in analyzing transient and/or non-stationary behavior. Recent advances in the field of wavelet theory show much promise in alleviating these problems. This thesis considers the realizations of the wavelet decomposition and reconstruction algorithms for the discrete case. The major discussion will involve both the one and two dimensional transforms. We also present a multiple-phase development as a second and possibly a preferable method for decomposing signals.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I thank my thesis advisor, Professor Lam, for being patient and pushing me in completing this thesis. I also appreciate all the technical support in both the Electrical and Computer Engineering SUN and PC labs, who all showed me the light in the picky world of the UNIX and PCDOS operating systems. Last but not least, I thank Kathy Ho for showing me that there is more to life than just studying.

I. INTRODUCTION

The recent advances of wavelet theory have spurned much interest in many disciplines: earthquake detection, medical EKG's, and multiple resolution signal processing. We are most interested in the last discipline listed, multiple resolution signal processing. Using wavelet theory in this field opens several areas of interest. In the one-dimensional case, analysis of short-time duration signals such as radar pulses, can possibly lead to better discrimination of similar but distinct radar sources. For the two-dimensional case, statistical pattern recognition using wavelets can aid in identifying skewed or rotated images. Lastly, for both dimensions, wavelet theory can be used in data compression, which has major applications particularly in transmitting data on a bandwidth-constrained channel. Traditionally, most means to date use Fourier methods to attack the problems in the areas of interest discussed previously. However, for transient and/or non-stationary phenomena, Fourier techniques give inadequate results in the transform domain, even when we modify them to include a dimension of time. Thus, wavelets can serve as an alternative to the conventional Fourier transform techniques for analyzing such transient and/or non-stationary behavior.

This thesis develops one and two-dimensional invertible discrete wavelet transforms (DWT). 386MATLAB and PROMATLAB Version 3.5 are the software tools used for 386 or better PC's, and for the UNIX-based SUN workstations. We did not use Fortran and C programming languages to maximize the portability of the routines, but such codes could be developed very easily from this thesis. These routines can be used by other researchers for more in-depth analysis of signals and systems.

Chapter II will cover the basic principles of the wavelet theory, particularly for the discrete dyadic case. Although many volumes are dedicated to the field, this cursory outline of the theory will provide the basis for notation used throughout the thesis. Chapter III will entail some important concepts such as causality and spillover effects that must be accounted for in the DWT. We expand these ideas further into the two-dimensional case in Chapter IV. Chapter V proposes another method for decomposing the data at the expense of physical memory, the so-called multiple-phase DWT (MP/DWT). The MP/DWT possibly approximates the continuous WT, with the property of time invariance. Chapter VI gives a narrative description of the MATLAB algorithms developed. Finally, we present our conclusions in Chapter VII.

II. BASIC WAVELET THEORY

Since wavelets are a relatively new and less established field to most readers, one main question asked is "what are they?" We introduce the topic in here by an example.

Consider a function f(t) that has discrete levels as shown in Figure 1. We decompose f(x) into a lower resolution level. In other words, we desire to smooth (or average) out f(x), or to low pass filter the function to a coarser level. Figure 2 depicts the resulting decomposition function ${}_{\alpha}f(x)$, where α is a scaling factor. For the dyadic case, we set α to two. If we compare the original function f(x) to the coarser function ${}_{\alpha}f(x)$, we note that the detail is ${}_{\alpha}d(x)=f(x)-{}_{\alpha}f(x)$ as shown in Figure 3. If we had the detail and the decomposed signal, we can theoretically reconstruct f(x) without any loss of information. The decomposition (and the reconstruction) process can go to any arbitrary resolution level **m**. Usually, we sample the signal at a rate higher than or equal to the Nyquist rate, and set the resulting sampled data to be at the highest resolution level (say level m=0). All other resolution levels would be in the negative direction. This process could be repeated for the next lower resolution level by operating on the current level in an iterated manner.

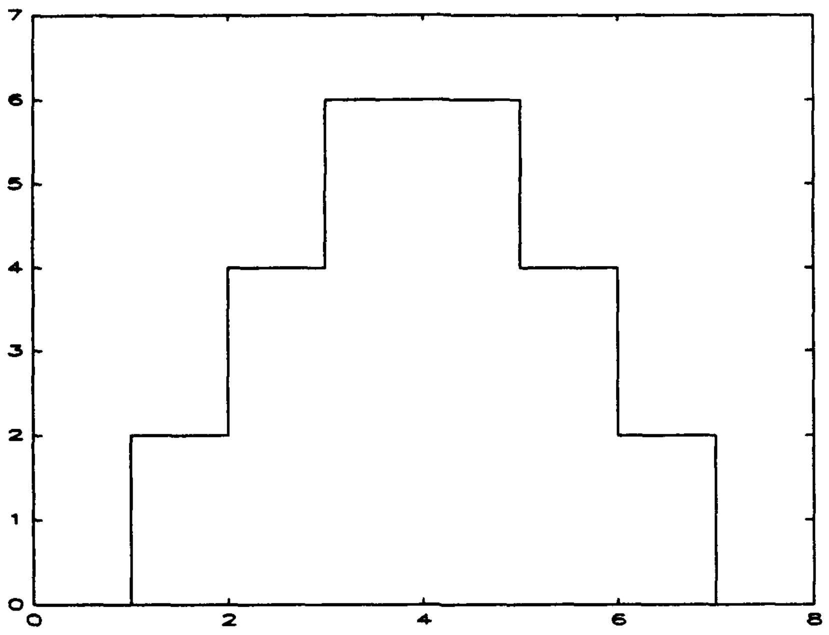
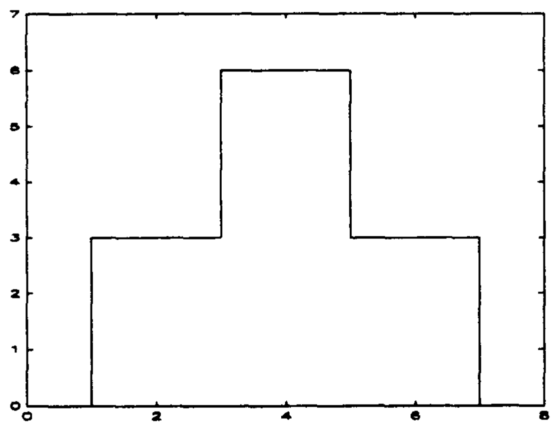


Figure 1. Example Function f(x)



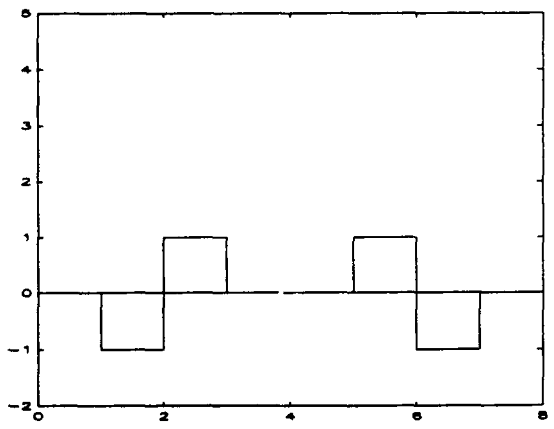


Figure 2. Blurred ${}_{\alpha}f(x)$ **Figure 3.** Detail ${}_{\alpha}d(x)$

The primary goal is to find a set of orthonormal basis functions that will do the smoothing of the original level and retain the details for reconstruction in a more practical manner. The low pass filtering process uses a function called the scaling function $\phi(x)$ while the detail basis function uses the wavelet basis function $\psi(x)$, both of which are also orthogonal to each other.

The described method is a coarse level in the understanding of the wavelet theory. We must now go to a higher resolution level of knowledge to better comprehend the following chapters.

A. THE SCALING FUNCTION $\phi(x)$

The region $L^2(R)$ depicts a vector space where a function f(x) resides and satisfies the condition of having finite energy, or the inner product of f(x) with its complex conjugate is finite. Consider a family of embedded closed subspaces, V_m, such that the infinitely largest subspace is $L^2(R)$ while the infinitely smallest subspace {0}.

$$\int_{-\infty}^{+\infty} f(x)f^*(x)\,dx = \langle f(x), f^*(x) \rangle < \infty \tag{1}$$

$$\text{(lower resolution } \ldots \subset V_{-1} \subset V_0 \subset V_{+1} \subset \ldots \text{ higher resolution)}$$

Additionally, for each vector space V_m in V_{m+1}, let W_m be an orthogonal complementary space to V_m in V_{m+1} such that $V_{m+1}=V_m \oplus W_m$. We show the spaces graphically in Figure 4.

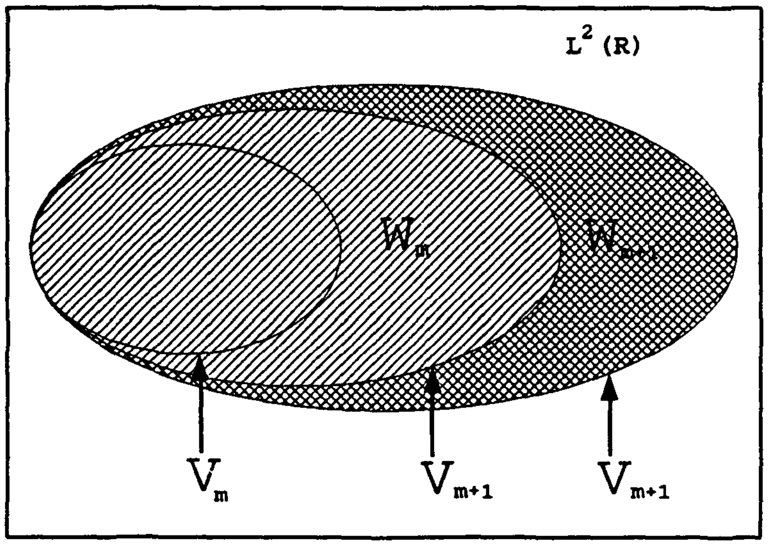


Figure 4. Vector Subspaces

Note that if given V_m and W_m to W_{m+k}, we can obtain the vector space V_{m+k+1} as shown in Figure 5.

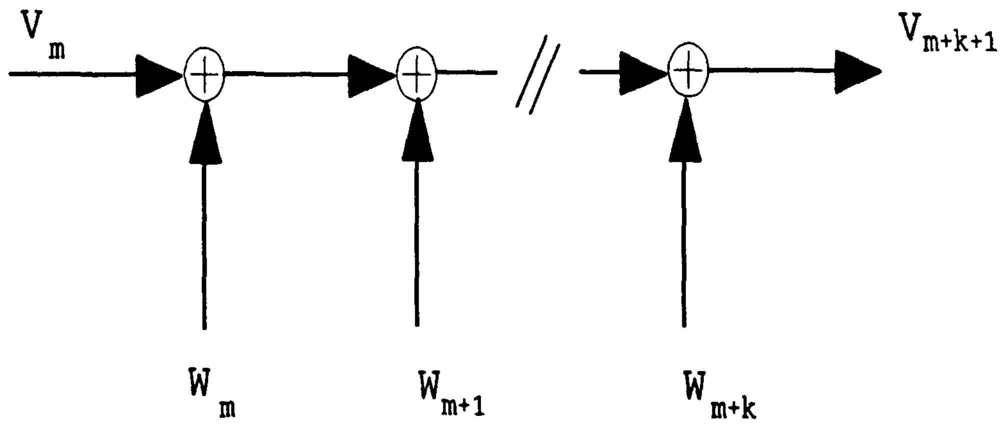


Figure 5. Another View

A function f(x) will lie in a vector space V_m if and only if it can be written as:

$$f_m(x) = \sum_{n \in Z} c_{mn} \, 2^{\frac{m}{2}} \, \phi(2^m x - n) \tag{2}$$

where we adjust the width of $\phi(x)$ for unit grid spacing at resolution level m=0. The inner product of ϕ with an n-shifted version of itself must equal zero ($<\phi(x),\phi(x\text{-}n)> = 0$).

In order to normalize the scaling function for the corresponding resolution level in the $L^2(R)$ space, multiply the scaling function by a factor $2^{m/2}$ while also dilating in the spatial domain. The following two equations simplify the notation for the rest of the derivations.

$$\phi_m = \phi_m(x) = 2^{\frac{m}{2}} \phi_0(2^m x) \tag{3}$$

$$\phi_{mn} = \phi_{mn}(x) = 2^{\frac{m}{2}} \phi_0(2^m x - n) = \phi_m(x - 2^{-m} n) \tag{4}$$

Let P_m be the orthogonal projection of a function in $L^2(R)$ onto the vector space V_m. Note that Equation 2 is a harmonic series representation of the function projected onto the V_m space, with c_{mn} denoting the weighting coefficients for the basis function. This is similar to the evaluation of the fourier coefficients for the fourier series expansion with its basis functions.

$$c_{mn} = \langle f(x) \, , \, \phi_{mn} \rangle \tag{5}$$

Thus Equation 5 can be viewed as a convolution of the function with the mirrored scaling function sampled every 2^m seconds.

$$c_{mn} = f(x) * \tilde{\phi}_m \;\Big|_{2^{-m}n} \tag{6}$$

The recursion property for the scaling function only depends on two adjacent resolution levels m and m+1. Since $V_m \subset V_{m+1}$, we have:

$$\phi_m = \sum_k \langle \phi_m , \phi_{m+1,k} \rangle \, \phi_{m+1,k} \tag{7}$$

We define the filter coefficients in terms of the inner product of Equation 7 as

$$h(k) = \int \phi_0(x) \; \phi_0(2x-k)dx = 2^{-\frac{1}{2}} \langle \phi_0 , \phi_{+1} \rangle \tag{8}$$

We are now able to bring the previous equations into a form known in the wavelet field as the *fundamental dilation equation.* This is a two scale difference equation relative to the two adjacent levels.

$$\phi(x) = \sum_k 2 \; h(k) \; \phi(2x-k) \tag{9}$$

A finite set of h(n) coefficients gives very desirable traits for selecting wavelets with compact support. Major research in the field has been in the study and determination of the family of scaling functions that are of compact support and orthonormal. With the scaling function being orthonormal, we define the following properties which are relative easy to prove [Ref. 1: pp. 689-691]:

$$\int \phi(x)\, dx = 1 \tag{10}$$

$$\sum_k h(k) = 1 \tag{11}$$

$$\sum_k h^2(k) = \frac{1}{2} \tag{12}$$

$$2 \sum_k h(k)\, h(k-2n) = \delta(n) \tag{13}$$

Going into the spectral domain, the low pass properties of the scaling function are obvious from the next four equations.

$$\Phi(f) = H\left(\frac{f}{2}\right) \Phi\left(\frac{f}{2}\right) \qquad \textit{where} \quad H(f) = \sum_k h(k) e^{-j2\pi kf} \tag{14}$$

$$\Phi(f) = \prod_{p=1}^{\infty} H\left(\frac{f}{2^p}\right) \tag{15}$$

$$\sum_k h(k) = 1 \quad \leftrightarrow \quad H(0) = 1 \tag{16}$$

$$|\, H(f)\, |^2 + |\, H(f+\frac{1}{2})\, |^2 = 1 \tag{17}$$

B. THE WAVELET BASIS FUNCTION $\psi(x)$

In a manner analogous to the development of the scaling function, the wavelet basis function $\psi(x)$ determines the W_m vector space. In a similar fashion to Equation 2, we define the detail function as follows:

$$d_m(x) = \sum_{n \in Z} d_{mn} \, 2^{\frac{m}{2}} \, \psi(2^m x - n) \qquad (18)$$

The wavelet basis function is of the form of a constant-Q band pass filter function, increasing an octave in bandwidth as the frequency increases an octave. Since W_m is orthogonal to V_m, the inner product of each of the spaces' basis functions should equal zero ($< \psi(x) , \phi(x) > = 0$). This relationship is so intertwined that by knowing a valid set of ϕ_{mn}'s, a set of ψ_{mn}'s can be derived for a particular resolution level. Consequently, knowing the h(n) coefficients will determine the appropriate coefficients for the wavelet basis functions given as g(n).

We define the wavelet basis in a similar fashion as the dilation property in Equation 9:

$$\psi(x) = 2 \sum_{k} g(k) \, \phi(2x - k) \qquad (19)$$

The $\{\psi_m(x-n)\}$ is an orthonormal set spanning W_m. The g(n) coefficients share many of the properties of the h(n) sequence with the following exceptions:

$$\int \psi(x)\,dx = \sum_k g(k) = 0 \tag{20}$$

$$\Psi(f) = G\left[\frac{f}{2}\right] \Phi\left[\frac{f}{2}\right] \tag{21}$$

Mallat [Ref. 1: p. 681] and Daubechies [Ref. 2: p. 944] chose the g(n)'s in terms of the h(n)'s as follows:

$$g(n) = (-1)^{1-n}\, h(1-n) \tag{22}$$

C. THE DISCRETE WAVELET TRANSFORM

Note that Equation 6 shows the lower resolution, or the approximation, signal's weighting coefficients as a convolution of the time reversal of the h coefficients with the signal f(x) sampled every 2^m seconds. Using the recursive properties of the scaling function, it is easily shown that the approximation coefficients of the lower resolution levels also can be determined from approximation coefficients of the higher level. In particular:

$$
\begin{aligned}
c_{mn} &= \langle f(x)\ ,\ \phi_{mn} \rangle \\
&= \int f(x) \sum_k \langle \phi_m, \phi_{m+1} \rangle\, \phi_{m+1} dx \\
&= \sum_k \langle \phi_m, \phi_{m+1} \rangle \int f(x) \phi_{m+1} dx \\
&= \sum_k \langle \phi_m, \phi_{mm+1} \rangle\ \ \langle f(x), \phi_{m+1} \rangle \\
&= \sum_k 2^{\frac{1}{2}}\ h(k)\ c_{m+1,n} \\
&= 2^{\frac{1}{2}} \sum_k h(k)\ c_{m+1}(2n+k) \\
&= 2^{\frac{1}{2}} \sum_j \tilde{h}(2n-j) c_{m+1}(j)
\end{aligned}
\qquad (23)
$$

where $\tilde{h}(n) \doteq h(-n)$.

The detail weighting coefficients can similarly be determined as

$$
d_{mn} = 2^{\frac{1}{2}} \sum_j \tilde{g}(2n-j) c_m(j) \qquad (24)
$$

Recall that the d_{mn}'s are the detail coefficients of the orthogonal projection of f(x) onto the W_m vector space. The entire collection of $\{d_{mn}\}$, or $\{c_{-Mn}\} \cup \{d_{mn} : m \geq -M\}$ is called the discrete wavelet transform.

1. Decomposition

Decomposing to any lower level requires that the $L^2(R)$ function be initially convolved with $\phi_0(-x)$, and sampled at a unity interval to obtain the coefficients, $c_0(n)$, as shown in Equation 2, and graphically depicted next:

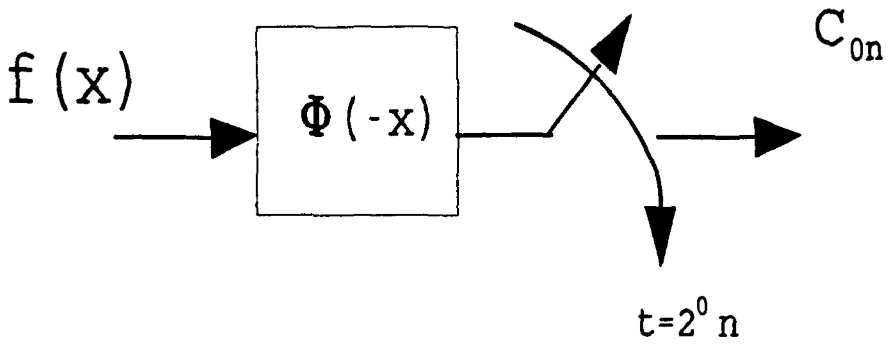


Figure 6. Determining m=0 Coefficients

Once the c_{0n} coefficients are known, we use Equations 23 and 24 in an iterated manner to obtain the set of coefficients for any arbitrary level. For the dyadic case, this process is a convolution, a two-times decimation of the result, followed by a $2^{1/2}$ scaling factor.

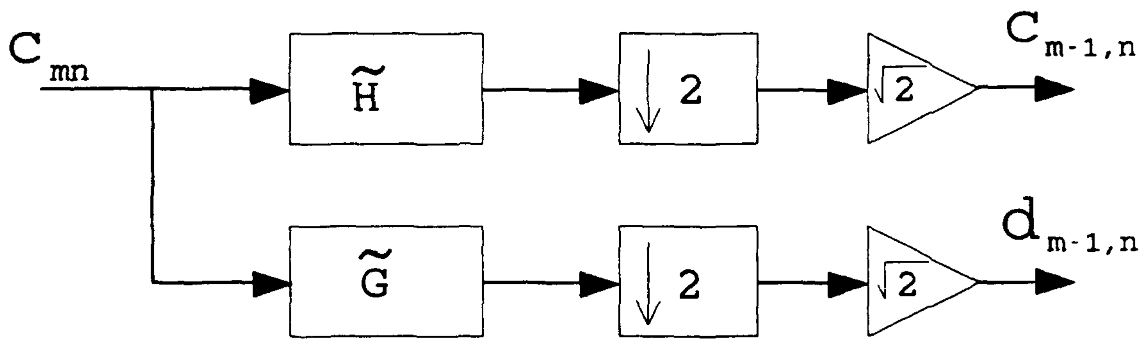


Figure 7. Decomposition Algorithm

2. Reconstruction

Recall that $V_{m+1}=V_m \oplus W_m$. The c_{nm}'s reside in the V_m space and the d_{nm}'s lie in the W_m space. Thus, a projection of a function onto the next higher level should equal the sum of the projections onto V_m and W_m. Let P_m and Q_m be the projection operators onto the vector space.

$$\begin{aligned} P_{m+1}\{f\} &= P_m\{f\} \oplus Q_m\{f\} \\ P_{m+1}\{f\} &= \sum_l c_{m+1}(l)\,\phi_{m+1}(x-2^{-(m+1)}l) \\ P_m\{f\} &= \sum_n c_m(n)\,\phi_m(x-2^{-m}n) \\ Q_m\{f\} &= \sum_n d_m(n)\,\psi_m(x-2^{-m}n) \end{aligned} \tag{25}$$

Matching the terms in Equation set 25 and doing several transformations of variables, we get the following:

$$c_{m+1} = \sqrt{2}\sum_l \left[c_m(l)\,h(n-2l) + d_m(l)\,g(n-2l)\right] \tag{26}$$

Equation 26 shows that the reconstruction of the approximation coefficients of one level higher can be realized by putting zeros between the lower resolution coefficients, convolving with the respective H or G filter, adding and then scaling the result by $2^{1/2}$. We show a block diagram of this basic algorithm in Figure 8.

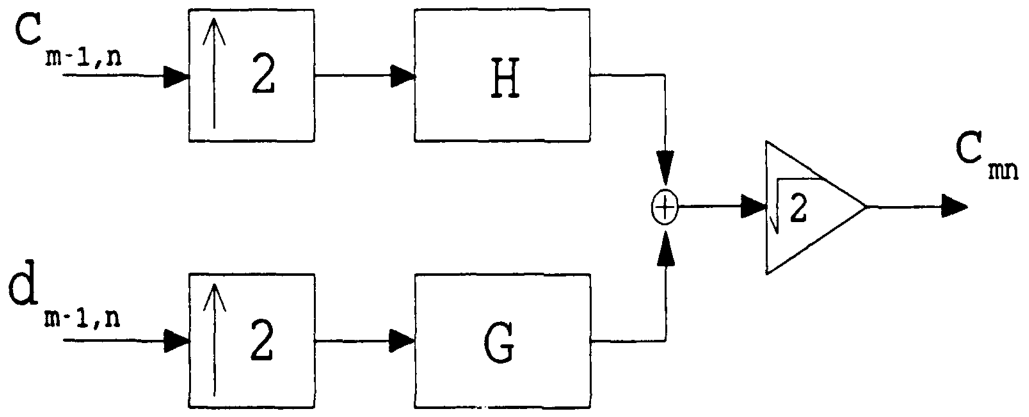


Figure 8. Reconstruction Algorithm

For the rest of the discussion, we will concentrate primarily on the approximation and detail coefficients. It is the properties of these coefficients that are of importance in relation to the DWT process.

The basic procedure would be to decompose the data array, starting from the highest resolution level. Next calculate the coefficients for the next lower level, while retaining the details, d_{mn}, and using the resulting approximation coefficients to calculate the next lower level. Repeat this procedure until reaching the lowest desired level where now both the approximation and detail coefficients are retained.

III. ONE DIMENSIONAL DWT DEVELOPMENT

A. DECOMPOSITION

1. Causality

Recalling that the decomposition algorithm of Figure 7, a finite data vector of the sampled signal, c_{0n}, can be decomposed to its lower resolution coefficients, c_{-1n} and d_{-1n}. This process requires that the data array be convolved with the respective filters. These filters are the time-reversed version of the H and G vector array. At the output of each convolution, there is a two times decimation. The $2^{1/2}$ scalar multiplication normalizes energy the in the $L^2(R)$ domain.

Consider a finite array c_{0n} of length L. Let c_{0n} to be a non-zero set only if the indices, n, satisfy $0 \leq n \leq L-1$. Also set the number of h coefficients equal to an even number N. By using Equation 23, the convolution process can be depicted as shifting the h's by a factor of two each time.

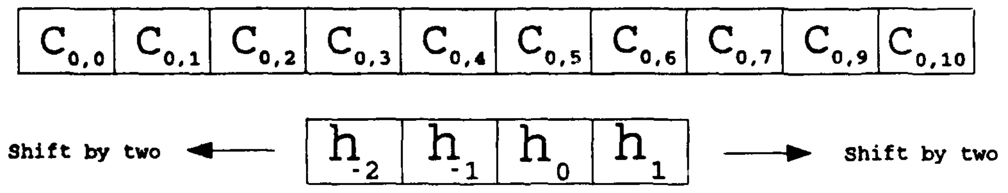


Figure 9. Shift Factor of 2 for N=4

In an effort to make the transform a causal system, we desire to make all the non-zero data coefficients have indices greater or equal to zero since negative indices would equate to negative time. This reconstruction gives two possible sets of indices for the h filter coefficients.

- {h(-N+1),...,h(-2),h(-1),h(0)}
- {h(-N+2),...,h(-1),h(0),h(1)}

We allow the negative indices for the filter since we already know the filter coefficients (a-priori data). These two sets for the filter coefficients equate to the two possible phases that can be used for the decomposition. For notation purposes, we will set the largest value of the indices to equate to the type of phase decomposition, phase-0 or phase-1.

By using Mallat and Daubechies selection of the g coefficients in Equation 22, there is a problem that occurs. While the resulting convolution with the h mirror filter gives the values for indices that are greater or equal to zero, the output of the detail coefficients give values for negative indices. In other words, given we define the h's with phase-0 indices, the resulting g coefficient indices are all greater or equal to one, as shown in Equation 23.

In order to get both the c_{nm}'s and d_{mn}'s to fall into the right hand side, we must find another relation for g in terms of h, that satisfies the properties in Equations 18-21. The follcwing equation will serve the purpose:

$$g(n) = (-1)^{-N+3} h(-N+3-n) \tag{27}$$

2. Phase Selection

The decomposition can proceed with the choice of phase to select. The selection of the desired decomposition phase and size of the input vector, L, determines the total number of coefficients at the lower resolution level.

$$\begin{aligned} phase\text{-}0: \quad \left|\left\{c^{0}_{m-1,n}\right\}\right| &= \left\lceil \frac{N+L-1}{2} \right\rceil \\ phase\text{-}1: \quad \left|\left\{c^{1}_{m-1,n}\right\}\right| &= \left\lfloor \frac{N+L-1}{2} \right\rfloor \end{aligned} \tag{28}$$

3. Zero-Padding of the Data Array

The basic algorithm uses the dot product between the h (and g) coefficient vector with a set of data coefficients, c_{mn}'s, that are obtained by a sliding window on the input coefficients, $c_{m+1,n}$ vector. The sliding window shifts to the right on a generated work vector to simplify the programming. This work vector is a zero-padded version of the input coefficient vector. Depending on the length of the input data array, the selection of the phase, and the size of the filters, the required zero-padding can be determined for the beginning and the end of the sequence.

If we select a phase-0 decomposition, we will generate the work array by initially padding one zero on the left. Additionally, if the size of the data array is even, we append another zero to the end of the sequence. For the phase-1 case, we pad the work array with one zero at the end of the sequence only if the original array is odd. After determining all these conditions, we finally pad the work array with N-2 zeros both in the beginning and the end of the intermediate sequence.

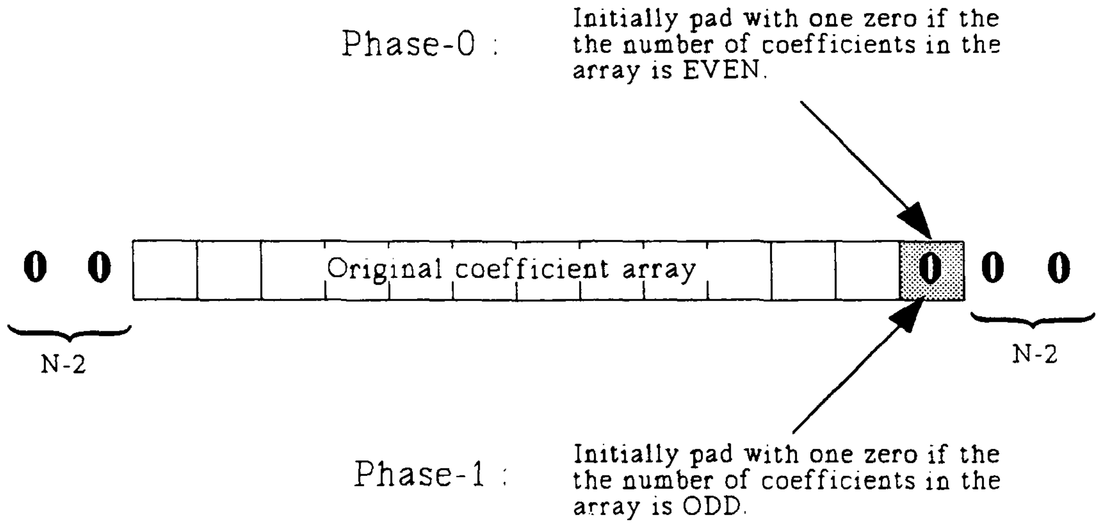

Phase-0 :
Initially pad with one zero if the the number of coefficients in the array is EVEN.
0 0
Original coefficient array
0
0 0
N-2
N-2
Phase-1 :
Initially pad with one zero if the the number of coefficients in the array is ODD.

Figure 10. Definition of the Work Array

In esoteric terms, a phase-1 decomposition for each level would be more efficient. Practically, this savings in memory is very insignificant, since the number of practical resolution levels is approximately $\lceil \log_2(| c_{00} |) \rceil$. Additionally, this reduction in the size of the workspace is balanced by a larger workspace during the reconstruction process. Most importantly, desiring only to decompose the data with only the phase-1 case for each level may miss some properties of a signal, which will be further discussed in Chapter V.

4. Energy Determination

One quick method in assessing if the decomposition routine is working without doing the reconstruction is to check the energy of the signal in each resolution level. Recall that Equations 2 and 18 are harmonic series representations. By using Poisson's Summation Formula (PSF), the energy in the function in the spatial domain is equal to the total energy stored in the weighting coefficients of the transform domain. This property holds true when the basis functions are orthonormal in the $L^2(R)$ space. Thus, the energy of a higher resolution level's approximation coefficients is equivalent to the sum of the energies of the approximation and detail coefficients one resolution level down.

$$\sum_n c_{m+1,n}^2 = \sum_l \left[c_{ml}^2 + d_{ml}^2\right] \tag{29}$$

By normalizing the energy to the input resolution level, m=0, a quick check of all the levels will add confidence to the program accuracy. In addition, this energy check shows the numeric truncation errors inherent on the processing unit used. The display of the energy will provide further insight on possible compression schemes for classes of signals, since the energy may be only concentrated at certain resolution levels.

Now as an example, consider a one-dimensional transient signal*, with a phase-1 decomposition for all the levels and using a Daubechies 2-tap filter (N=4) [Ref. 2: p. 980]. The following diagrams show the original signal and the energy plot.

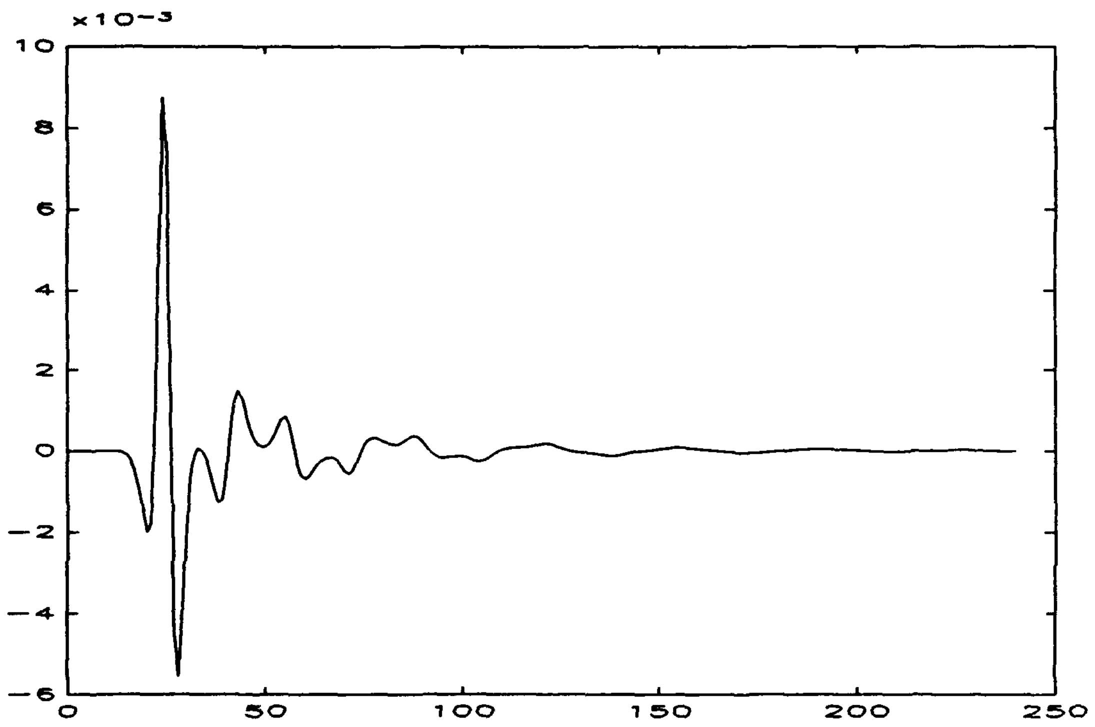


Figure 11. Transient Signal

* Professor Michael A. Morgan, Chairman of the Department of Electrical and Computer Engineering at the Naval Postgraduate School, provided the transient data included in the MATLAB routines called "et90.m ." This signal is the back-scattered transient electromagnetic field from a 10cm long thin-wire illuminated by a double-Gaussian impulse, computed using a time-domain integral equation. The other transient signal provided by Professor Morgan is "et60.m". The numbers 90 and 60 correspond to the angle of incidence that the double-Gaussian impulse impinges onto the wire's axis.

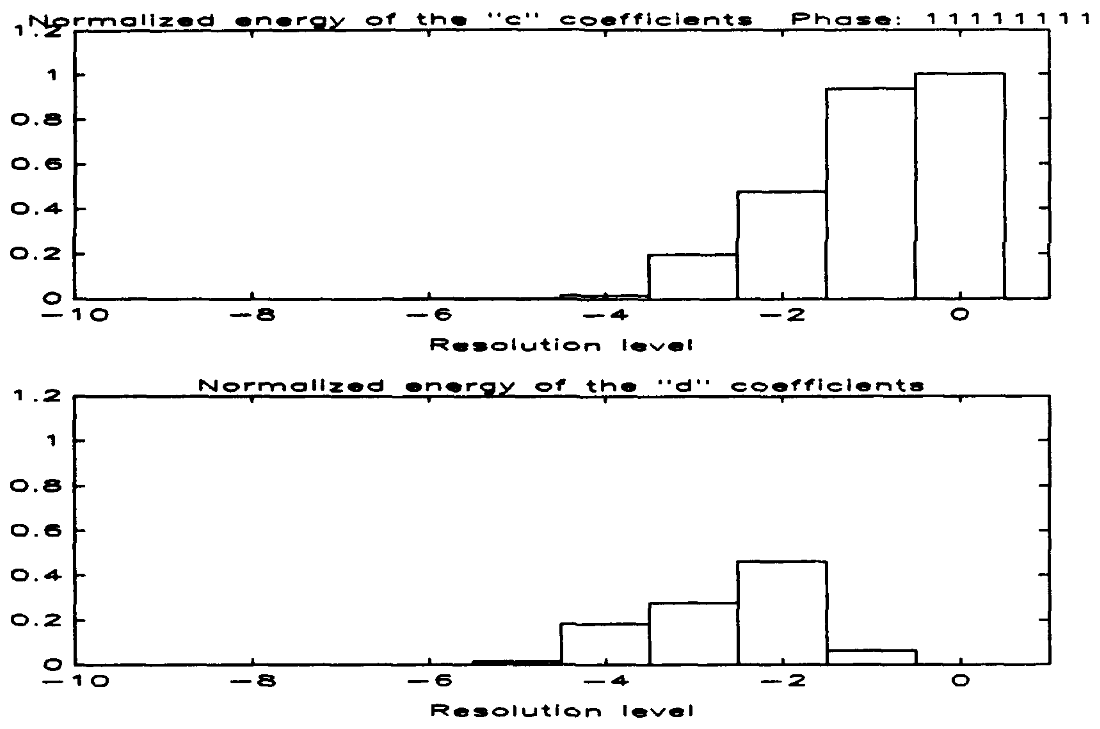


Figure 12. Energy in each Level

The energy check going up from the lowest resolution is depicted next. Note that the Daubechies h filter coefficients are only significant to 10^{-12} places, thus they cause insignificant numerical errors in the reconstruction.

Verification of the energy in each resolution level

Level	Energy in level m-1 c+d	Energy in C level m	Difference
-7	0.000024166621151	0.000024166621151	0.000000000000000
-6	0.000258409042490	0.000258409042489	0.000000000000000
-5	0.001707402900115	0.001707402900114	0.000000000000001
-4	0.015878270678385	0.015878270678372	0.000000000000013
-3	0.197762381708774	0.197762381708634	0.000000000000140
-2	0.474398688169256	0.474398688168927	0.000000000000329
-1	0.936969604304463	0.936969604304102	0.000000000000361
0	1.000000000000806	1.000000000000000	0.000000000000806

5. Approximation and Detail Time/Scale Diagrams

One interesting result of the DWT for the one dimensional case is the display of the energy on the time/scale plane. The scale can easily be related to frequency by noting that the highest resolution is the sampling frequency at m=0. As the scale decreases one level, we halve the center frequency for this scale.

Figure 7 shows the decimation of the coefficients by a factor of two. When we obtain the output of the decomposition process, the coefficient's are in a packed format, meaning that the appropriate time spacing is not preserved. Thus, 2^{-m}-1 zeros must be padded between each coefficient for a particular resolution level m.

The following time/frequency diagrams show the transient signal, mentioned in the previous section, with the proper spacing of the coefficients' energy in the transformed domain.

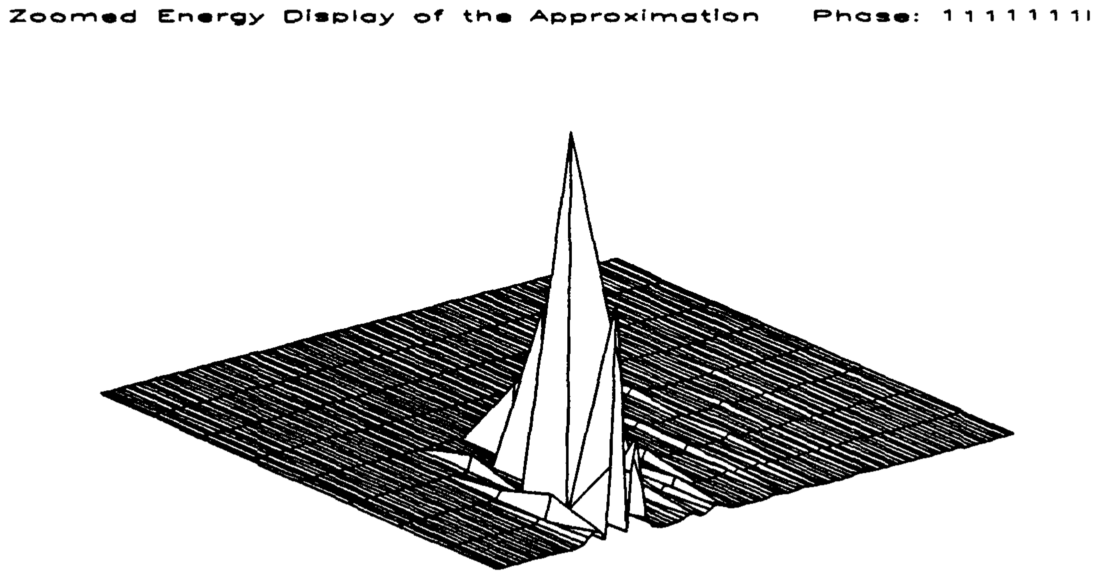


Figure 13. Mesh Plot for the Energy of the Approximation Coefficients to Eight Resolution Levels

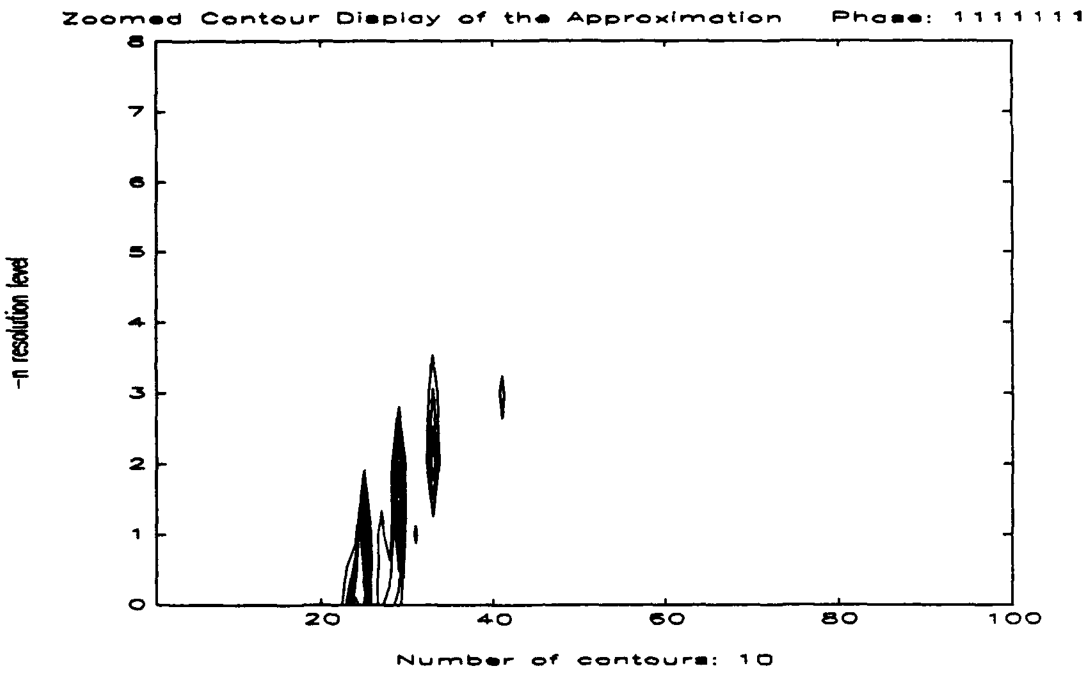


Figure 14. Contour Display for the Energy of the Approximation Coefficients to Eight Resolution Levels

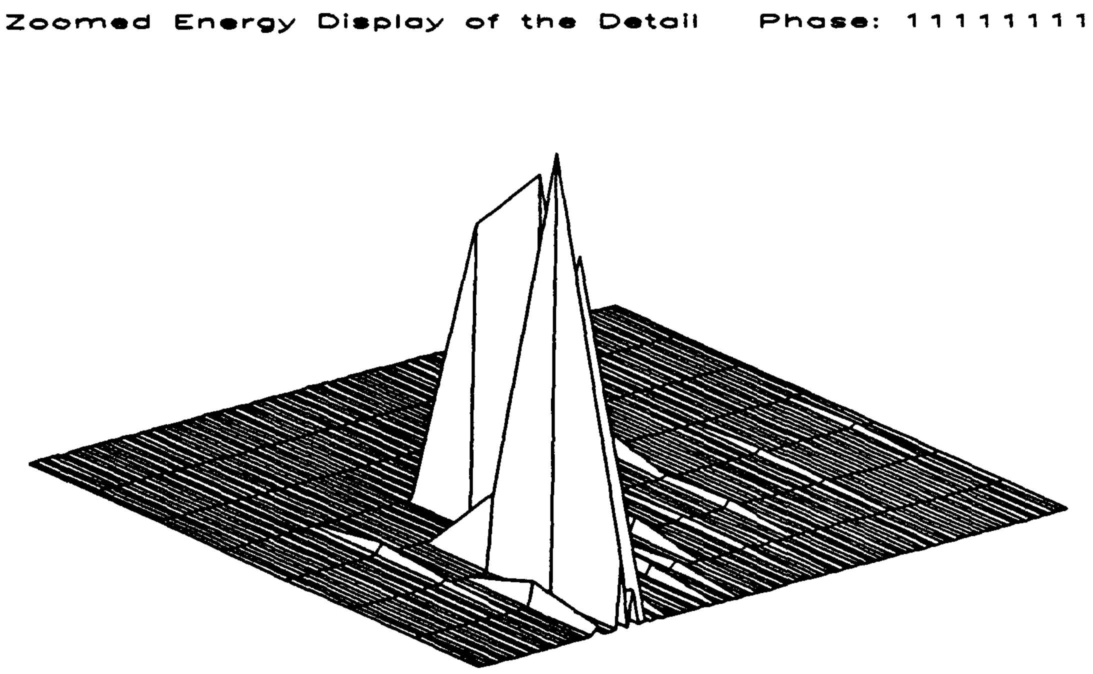


Figure 15. Mesh Plot for the Energy of the Detail Coefficients to Eight Resolution Levels

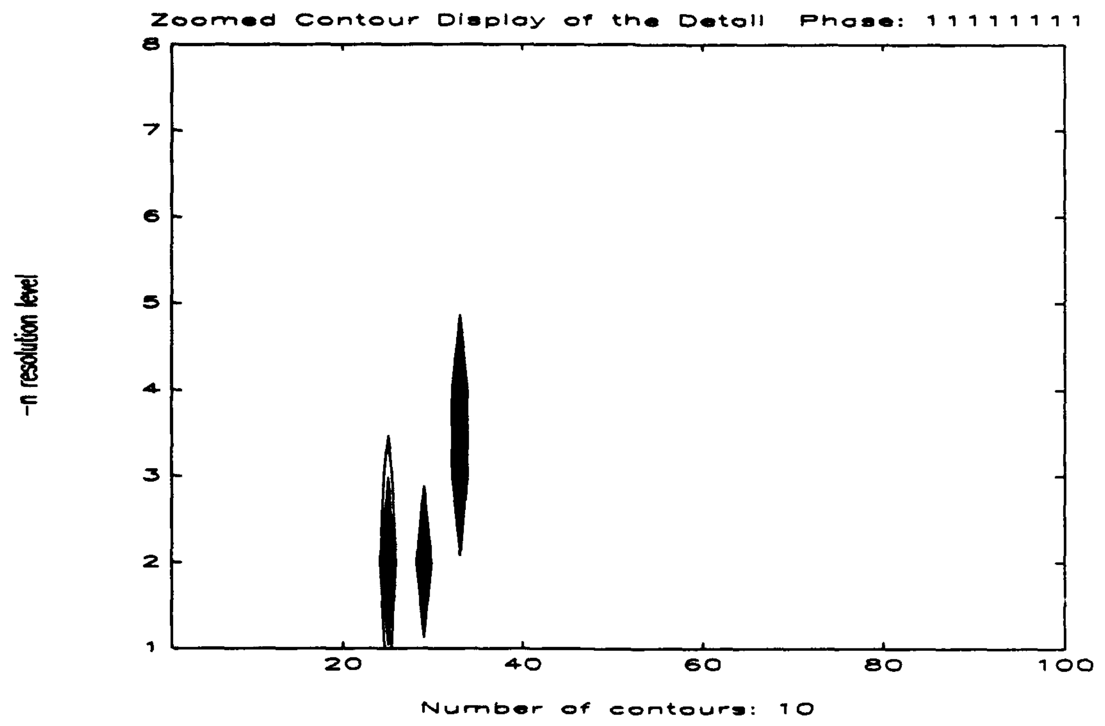


Figure 16. Contour Display for the Energy of the Detail Coefficients to Eight Resolution Levels

B. RECONSTRUCTION

Figure 8 and Equation 25 show the realization of the recomposition routine by first inserting a zero after each coefficient. Next we convolve the array with their respective filter, sum, and finally normalize by a factor of $2^{1/2}$. This process is essentially an interpolation.

Since we have defined the indexing of the h and g coefficients during the decomposition phase, the recomposition process is fixed to one of two methods depending on the decomposition phase used to get to that resolution level. The same dyadic convolutional algorithm can be used for the recomposition after initially massaging the input data arrays, and shifting one unit to the right instead of two. Initially, the h's and the g's are reversed in order.

Now we generate the workspace vectors for the respective coefficients. Append a zero AFTER each of the input coefficient vectors. Then add two more zeros at the end of the modified work vectors. Finally, if the phase-1 decomposition process was utilized to obtain the coefficients, append another zero at the beginning of the modified sequence. The diagram below summarizes the work vector definition prior to running the convolution algorithm.

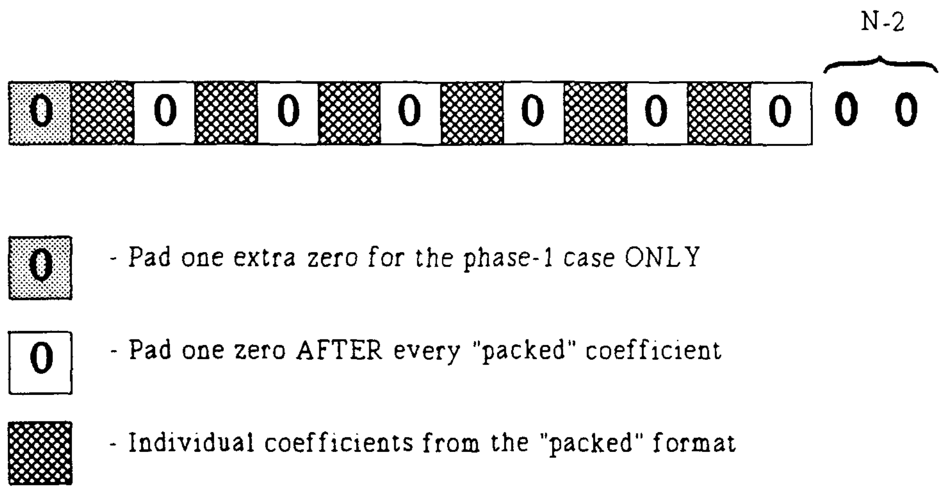

N-2
0 0 0 0 0 0 0 0 0
- Pad one extra zero for the phase-1 case ONLY
- Pad one zero AFTER every "packed" coefficient
- Individual coefficients from the "packed" format

Figure 17. One-dimensional Reconstruction Work Array

All the information needed for the reconstruction is the selected phase decomposition to get the c's and d's for that resolution level. One factor to note is that with Equations 26, there is a possibility that an extra coefficient be generated in the reconstruction process. This extra coefficient will theoretically equal zero, but numerically it may not and can be cascaded into a notable error if not accounted for. So since we know the length of the input vector at resolution level m=0, the size of the data array will be known for all levels since we retain all the detail coefficients. Any extra zero-valued coefficient generated can be identified as such and ignored in going up to the next resolution level.

Errors in the regeneration will be totally manifested as numeric errors of the computer processors. The following diagram shows the reconstructed transient signal and absolute error with the original.

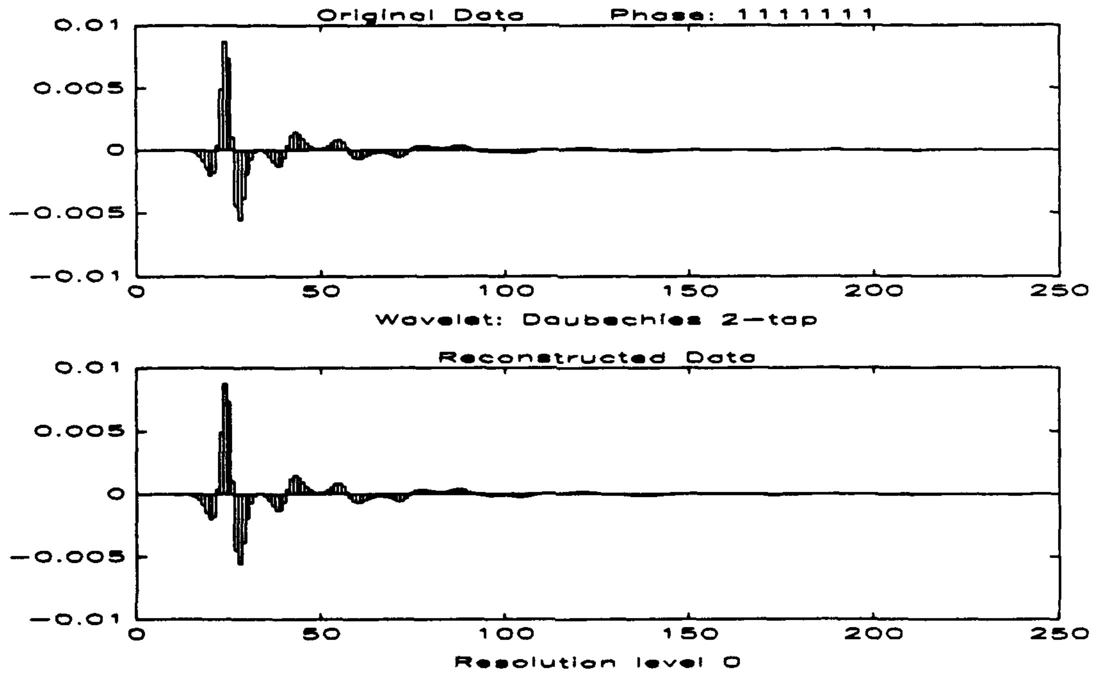


Figure 18. Original Transient Signal and Reconstructed Versions

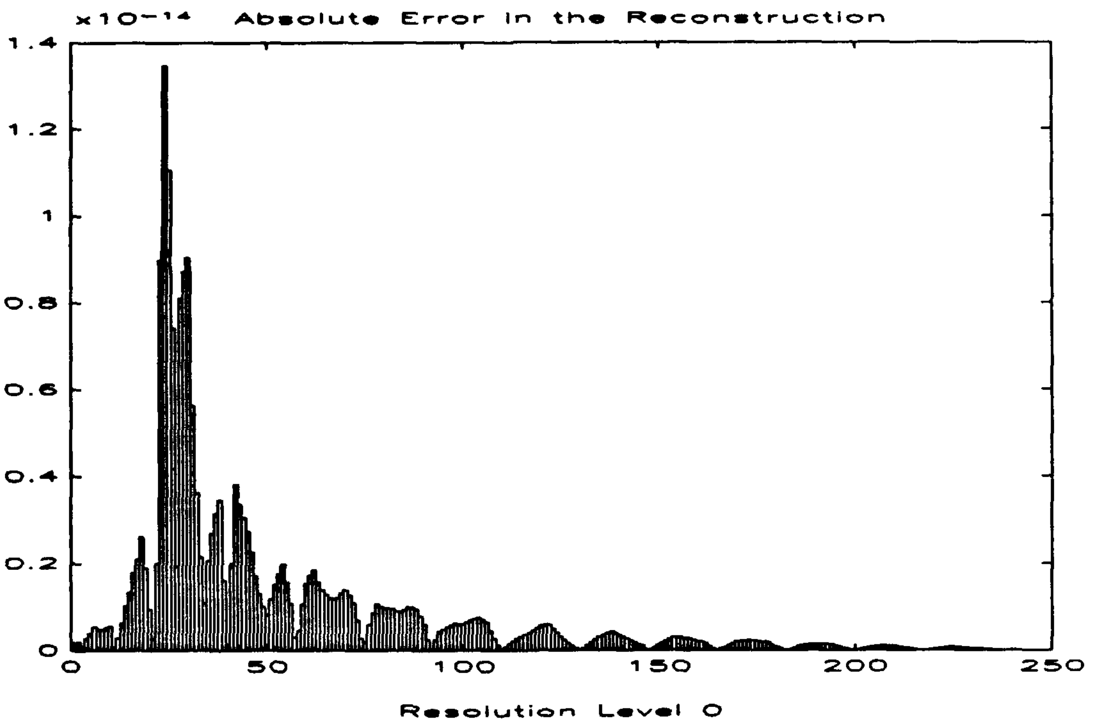


Figure 19. Magnitude of the Absolute Error of the Reconstruction. Notice the maximum value.

IV. TWO DIMENSIONAL DWT DEVELOPMENT

A. INTRODUCTION

Arguments for the DWT can be generated for any higher dimension. Of particular concern is the two-dimensional case for image processing. The vector subspaces now will reside in the $L^2(R^2)$ vice $L^2(R)$ space. A two-dimensional scaling function $\phi(x,y)$ exists whose scaled dilations and translations for a particular resolution level form an orthonormal basis for the vector subspace V_m. By making the two-dimensional scaling function separable, $\phi(x,y)=\phi(x)\phi(y)$, each vector space can be decomposed as a tensor product of two identical subspaces in $L^2(R)$.

$$V_m = V_m^1 \otimes V_m^1 \tag{30}$$

The tensor product can be viewed as the in-phase and quadrature-phase components in V_m. The properties depicted in Figures 4 and 5 still apply for the two-dimensional case.

$$\begin{aligned} V_{m+1} &= V_{m+1}^1 \otimes V_{m+1}^1 \\ &= \left(W_m^1 \oplus V_m^1\right) \otimes \left(W_m^1 \oplus V_m^1\right) \\ &= \left(V_m^1 \otimes V_m^1\right) \oplus \left(V_m^1 \otimes W_m^1\right) \oplus \left(W_m^1 \otimes V_m^1\right) \oplus \left(W_m^1 \otimes W_m^1\right) \end{aligned} \tag{31}$$

Notice that ϕ_{mn} defines a set of orthonormal basis functions in $V_{m+1}{}^1$ and ψ_{mn} in $W_m{}^1$. Thus, Equation 31 implies that four sets of equations form the orthonormal basis of V_{m+1} we summarize in the following table.

Basis Function	Vector Space	Basis Coefficient	Frequency Characteristics
$\phi(x)_{nm}\ \phi_{mn}(y)$	$V_m{}^1 \otimes V_m{}^1$	$c_m(k,l)$	Vert Low Pass Horiz Low Pass
$\phi(x)_{nm}\ \psi_{mn}(y)$	$V_m{}^1 \otimes W_m{}^1$	${}^1d_m(k,l)$	Vert Low Pass Horiz Band Pass
$\psi(x)_{nm}\ \phi_{mn}(y)$	$W_m{}^1 \otimes V_m{}^1$	${}^2d_m(k,l)$	Vert Band Pass Horiz Low Pass
$\psi(x)_{nm}\ \psi_{mn}(y)$	$W_m{}^1 \otimes W_m{}^1$	${}^3d_m(k,l)$	Vert Band Pass Horiz Band Pass

Let P_m, ${}^1D_m$, ${}^2D_m$, and ${}^3D_m$ denote the projection of a $L^2(R^2)$ function f(x,y), onto vector subspaces $V_m \otimes V_m$, $V_m \otimes W_m$, $W_m \otimes V_m$, and $W_m \otimes W_m$ respectively. Then in a fashion analogous to Equation 25, the vertical and horizontal low pass operation of f(x,y) onto resolution level m is as follows:

$$P_m f = \left(f(x,y) ** \ \phi_{mk}\phi_{ml}\right) = 2\sum_{k,l\,\in\,Z^2}\sum c_m(k,l)\,\phi_m(2^m x-k)\,\phi_m(2^m y-l)) \qquad (32)$$

Notice that the weighting coefficients, c_m, are now indexed in two-dimensions. Similarly, the ${}^1d_m$, ${}^2d_m$, and ${}^3d_m$ coefficients can be determined which correspond to the low horizontal and high vertical, high horizontal and low vertical, and high horizontal and high vertical frequency details.

Since the three two-dimensional wavelets and the one low pass basis functions are separable, the two-dimensional case is very easily realized. The weighting coefficients can be determined by first decomposing on the rows, then the columns of the array, or vice versa. The general algorithm is depicted in Equations 33-36 and Figure 20.

$$c_{m-1}(i,j) = 2\sum_k \sum_l c_m(k,l)h_h(l-2j)h_v(k-2i) \tag{33}$$

$$^1d_{m-1}(i,j) = 2\sum_k \sum_l c_m(k,l)h_h(l-2j)g_v(k-2i) \tag{34}$$

$$^2d_{m-1}(i,j) = 2\sum_k \sum_l c_m(k,l)g_h(l-2j)h_v(k-2i) \tag{35}$$

$$^3d_{m-1}(i,j) = 2\sum_k \sum_l c_m(k,l)g_h(l-2j)g_v(k-2i) \tag{36}$$

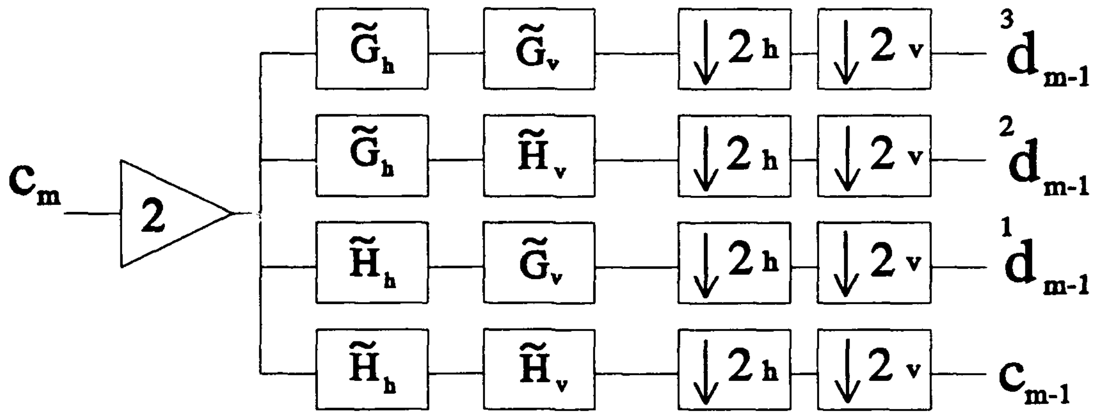


Figure 20. 2-D Decomposition Routine

B. DECOMPOSITION

We will use the convention in image processing that the upper left corner is the starting and the lower right is the ending data point. This way, we extend the idea of causality into two dimensions, where the spillover effects will fall to the right and below the input array.

1. Decomposition Mask

By looking at the ${}^1d_{m-1}$ coefficients' indices between the left and right side of the equation, generate a two-dimensional mask by forming the outer product of the vertical and horizontal filter coefficients.

$$H_h G_v = [g_v]^T \cdot [h_h] \tag{37}$$

where $[h]=[h(-N+2) \ldots h(0)\ h(1)]$. Similarly the masks for H_hH_v, G_hH_v, and G_hG_v also can be easily determined.

We require four masks to decompose successfully an image as suggested in Figure 20 to generate the coefficients c_{m-1}, ${}^1d_{m-1}$, ${}^2d_{m-1}$, and ${}^3d_{m-1}$. H_hH_v, H_hG_v, G_hH_v, and G_hG_v are the corresponding filter masks as shown next:

$h_h(-2)h_v(-2)$	$h_h(-1)h_v(-2)$	$h_h(0)h_v(-2)$	$h_h(1)h_v(-2)$
$h_h(-2)h_v(-1)$	$h_h(-1)h_v(-1)$	$h_h(0)h_v(-1)$	$h_h(1)h_v(-1)$
$h_h(-2)h_v(0)$	$h_h(-1)h_v(0)$	$h_h(0)h_v(0)$	$h_h(1)h_v(0)$
$h_h(-2)h_v(1)$	$h_h(-1)h_v(1)$	$h_h(0)h_v(1)$	$h_h(1)h_v(1)$

Figure 21. Mask H_hH_v

$h_h(-2)g_v(-2)$	$h_h(-1)g_v(-2)$	$h_h(0)g_v(-2)$	$h_h(1)g_v(-2)$
$h_h(-2)g_v(-1)$	$h_h(-1)g_v(-1)$	$h_h(0)g_v(-1)$	$h_h(1)g_v(-1)$
$h_h(-2)g_v(0)$	$h_h(-1)g_v(0)$	$h_h(0)g_v(0)$	$h_h(1)g_v(0)$
$h_h(-2)g_v(1)$	$h_h(-1)g_v(1)$	$h_h(0)g_v(1)$	$h_h(1)g_v(1)$

Figure 22. Mask H_hG_v

$g_h(-2)h_v(-2)$	$g_h(-1)h_v(-2)$	$g_h(0)h_v(-2)$	$g_h(1)h_v(-2)$
$g_h(-2)h_v(-1)$	$g_h(-1)h_v(-1)$	$g_h(0)h_v(-1)$	$g_h(1)h_v(-1)$
$g_h(-2)h_v(0)$	$g_h(-1)h_v(0)$	$g_h(0)h_v(0)$	$g_h(1)h_v(0)$
$g_h(0)h_v(1)$	$g_h(-1)h_v(1)$	$g_h(0)h_v(1)$	$g_h(1)h_v(1)$

Figure 23. Mask G_hH_v

$g_h(-2)g_v(-2)$	$g_h(-1)g_v(-2)$	$g_h(0)g_v(-2)$	$g_h(1)g_v(-2)$
$g_h(-2)g_v(-1)$	$g_h(-1)g_v(-1)$	$g_h(0)g_v(-1)$	$g_h(1)g_v(-1)$
$g_h(-2)g_v(0)$	$g_h(-1)g_v(0)$	$g_h(0)g_v(0)$	$g_h(1)g_v(0)$
$g_h(-2)g_v(0)$	$g_h(-1)g_v(1)$	$g_h(0)g_v(1)$	$g_h(1)g_v(1)$

Figure 24. Mask G_hG_v

These masks generate the corresponding coefficients by shifting by a factor of two to the right and downward through a workspace array. With the one-dimensional case, there were only two possible phase decompositions. Ncw four possible phases exist. Obviously, the workspace array is different for each phase decomposition.

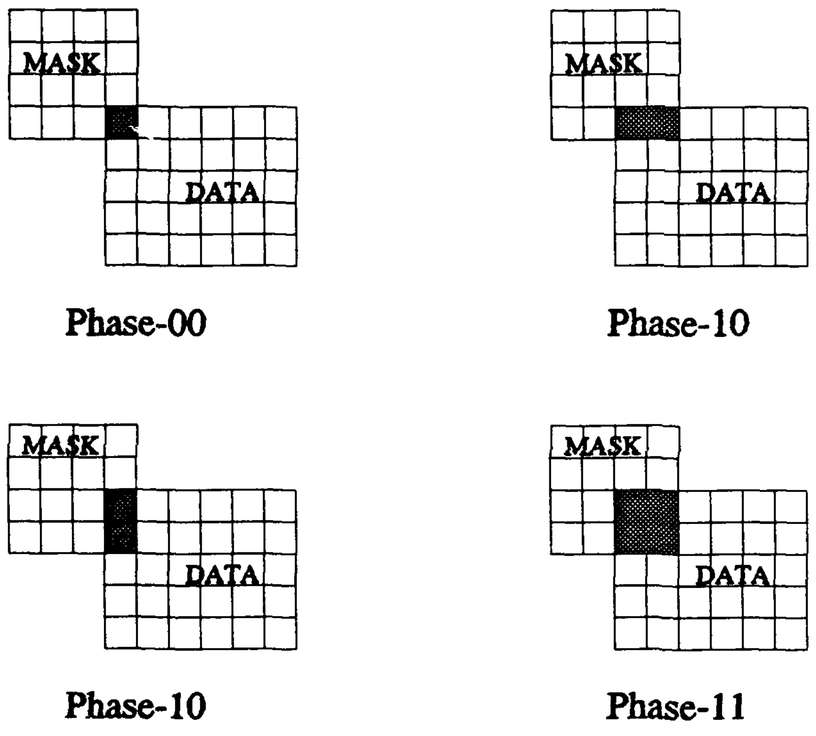
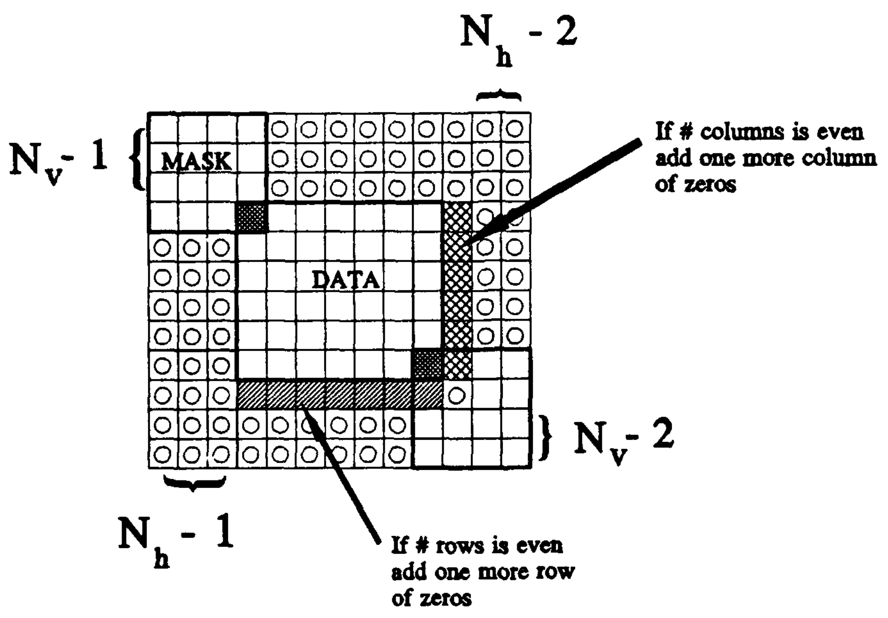


Figure 25. The Four Possible 2-D Decomposition Phases

2. Zero-padding of the Data Array

As stated from the previous section, the work space arrays are different from each other depending on one of the four decomposition phases selected, the size of the mask, and the size of the input array. These considerations must be noted.

The generation of the work array for the phase-00 case first consists of determining if the input array rows and/or columns are odd or even. If either of them are even, append a corresponding extra zeros row or column to the bottom or right of the input array, respectively. Given the number of rows and columns for the mask as N_v and N_h, add N_v-1 zeros rows above and N_v-2 zeros rows below the modified input array. Append N_h-1 zeros rows to the left and N_h-2 zeros rows to the right of the modified input array.



N_h - 2
N_v- 1
MASK
DATA
If # columns is even add one more column of zeros
N_v- 2
N_h - 1
If # rows is even add one more row of zeros

Figure 26. Phase-00 Work Array

For the phase-10 case, if the original array's rows are even and/or if the columns are odd, add one respective zeros row/column to the corresponding bottom/right of the modified array. Next add N_v-1 zeros rows above and N_v-2 zeros rows below the work array. Lastly, add N_h-2 zeros columns to the right and left of the array.

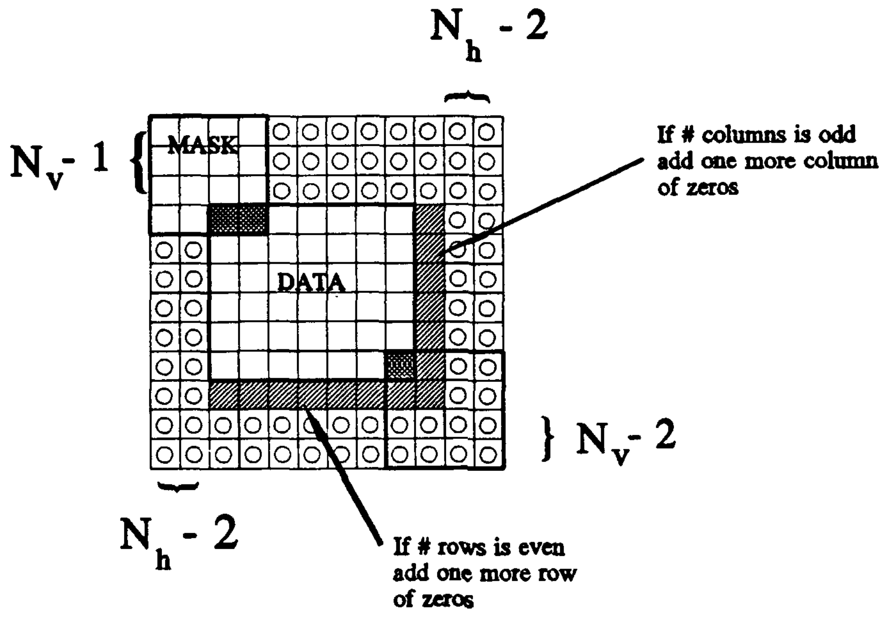

N_h - 2
N_v - 1
MASK
DATA
If # columns is odd add one more column of zeros
N_v - 2
N_h - 2
If # rows is even add one more row of zeros

Figure 27. Phase-10 Work Array

Alternatively the phase-01 case, if the original array's rows are odd and/or if the columns are even, add one respective zeros row/column to the corresponding bottom/right of the modified array. Next add N_v-2 zeros rows above and below the work array. Lastly, add N_h-1 zeros columns to the right and N_h-2 zero columns to the left of the array.

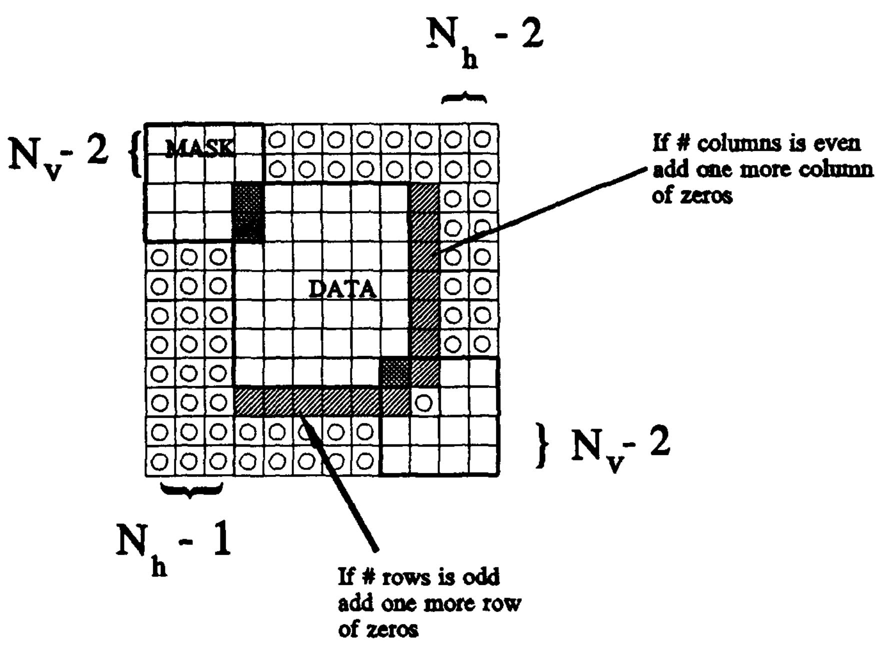

$N_h - 2$
$N_v - 2$
MASK
DATA
If # columns is even add one more column of zeros
$N_v - 2$
$N_h - 1$
If # rows is odd add one more row of zeros

Figure 28. Phase-01 Work Array

Lastly for the phase-11 case, if either of the input array's rows or columns are odd, then include a corresponding extra zeros row or column to the bottom or right of the input array, respectively. Given the number of rows and columns for the mask as N_v and N_h, add N_v-2 zeros rows above and below the modified input array. Add N_h-2 zeros rows to the left and to the right of the modified input array.

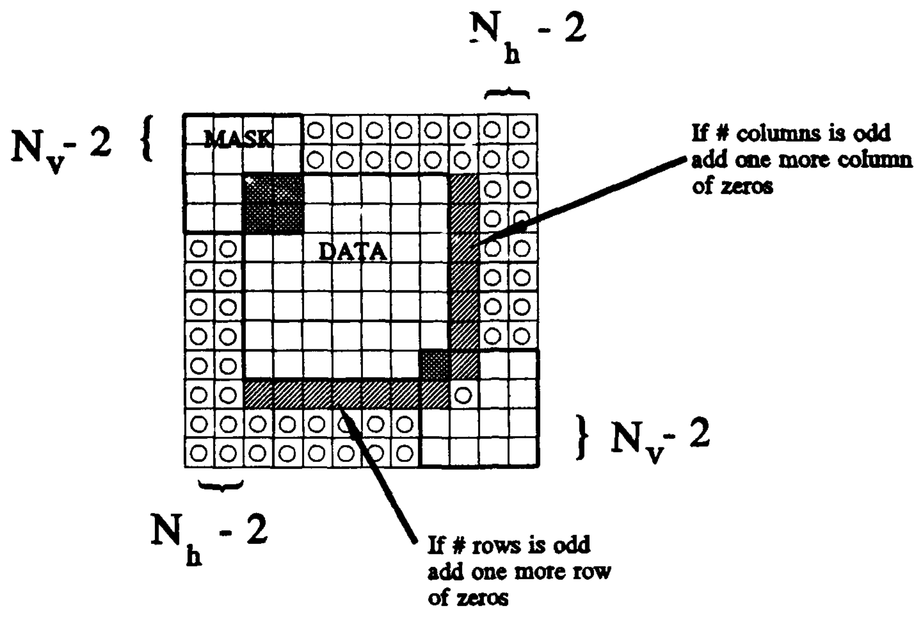

$N_h - 2$
$N_V - 2$
MASK
DATA
If # columns is odd add one more column of zeros
$N_V - 2$
$N_h - 2$
If # rows is odd add one more row of zeros

Figure 29. Phase-11 Work Array

3. Energy Determination

Similar to the one-dimensional case, adding up the squares of the four sets of coefficients at resolution level m will equal the energy in the coefficients for the next higher level, m+1. This is a natural two-dimensional extension of Poisson's Summation Formula, discussed previously in Chapter III.A.5, as shown as follows:

$$\sum_j \sum_k |c_{m+1}|^2 = \sum_j \sum_k \left(|c_m|^2 + |{}^1d_m|^2 + |{}^2d_m|^2 + |{}^3d_m|^2 \right) \qquad (38)$$

As an example, consider an image of a centered square, in Figure 30, decomposed down to three resolution levels with a phase-00 HAAR wavelet. The energy plot in Figure 31 shows the distribution of the c's and the four d's for each level.

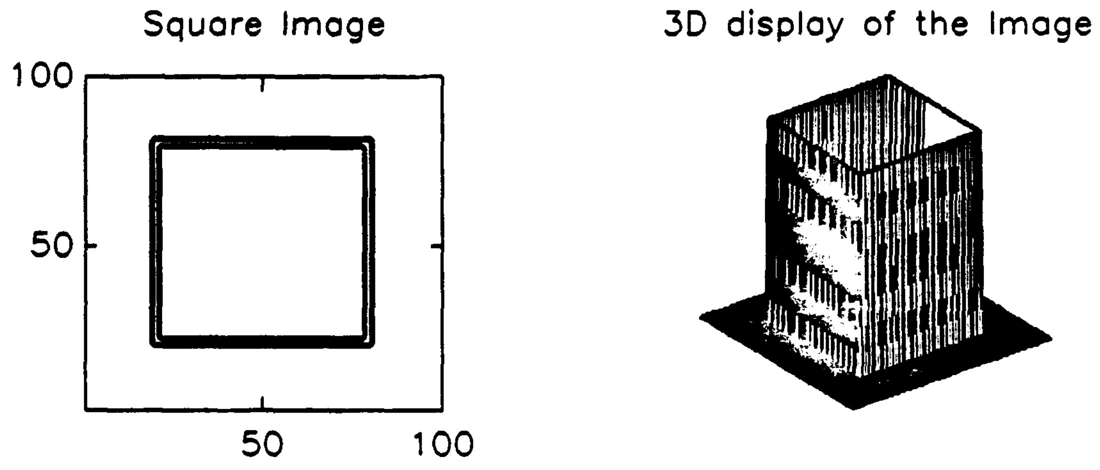


Figure 30. Square Image

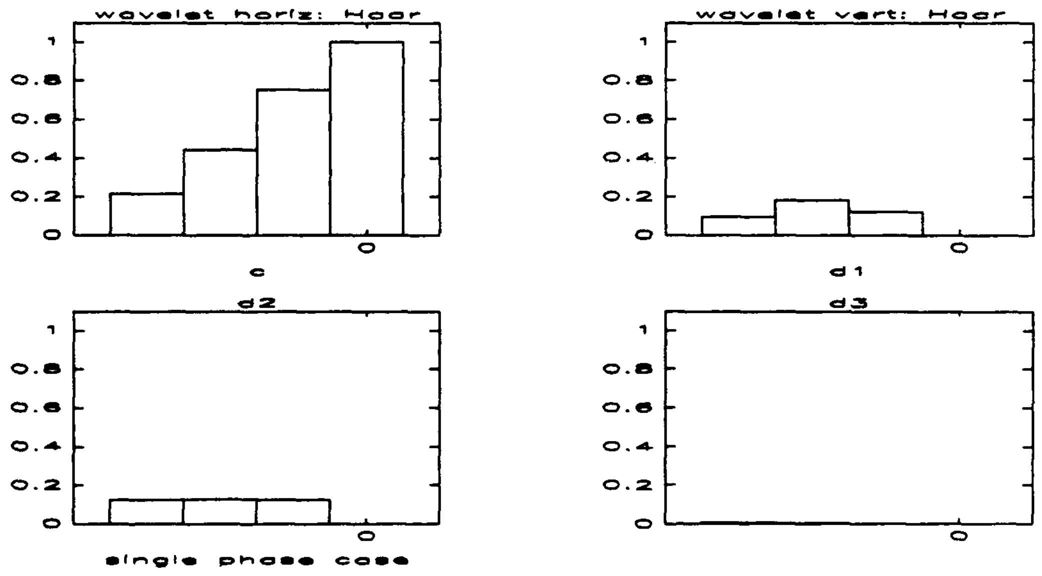


Figure 31. Energy Distribution

Except for numerical computations of the computer processor, all the energy from the lower d coefficients plus the energy of the lowest c coefficients add up to the energy of the original coefficients, c_0. The following energy check results for the two-dimensional case, normalized to c_0. Notice that there are no errors since we used Haar h filter coefficients and that the two-dimensional decomposition uses a normalization factor of two vice the square root of two.

```
         Verification of the energy in each resolution level

  Level      Energy in              Energy in C            Difference
    m        level m-1                level m
             c+d1+d2+d3
  --------------------------------------------------------------------
  -2      0.440942796610169     0.440942796610169     0.000000000000000
  -1      0.751059322033898     0.751059322033898     0.000000000000000
   0      1.000000000000000     1.000000000000000     0.000000000000000
```

C. RECONSTRUCTION

Since the scaling and wavelet basis functions are separable, the reconstruction algorithm closely follows the same process as in the one-dimensional case. First each row/column is zero padded appropriately and reconstructed with the one-dimensional technique. Then process each resulting column/row in a similar fashion but with a one unit shift to the right and then downward, as in a raster scan. We show this process in Figure 32.

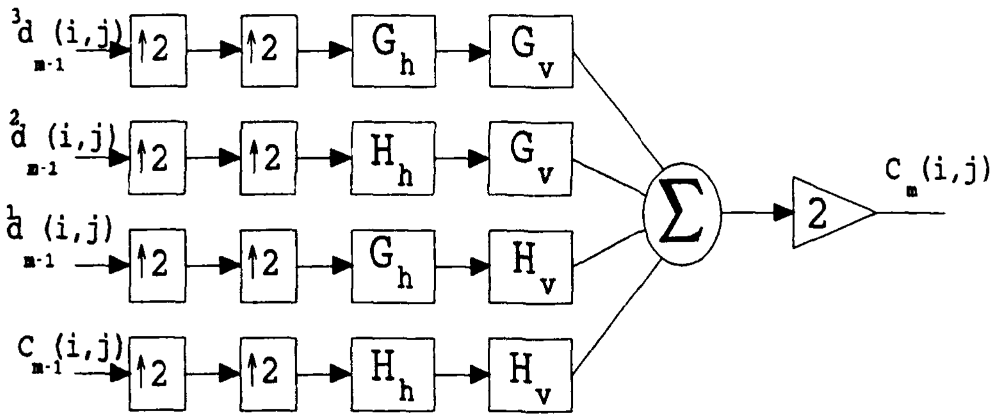


Figure 32. 2-D Reconstruction

By taking another look at Figure 32, a more efficient method can be realized. By padding each coefficient in the array with one zero in both the horizontal and vertical directions, we can then slide a reconstruction mask through the padded array.

1. Reconstruction Mask

We generate the reconstruction masks in a similar fashion as in the decomposition case, but with the important note that the coefficients are in the reverse order to accommodate the reconstruction convolution depicted in Figure 32 and the following equation in matrix notation

$$G_v H_h = [\hat{h}_h]^T \cdot [\hat{g}_v] \tag{39}$$

where $\hat{h}=[h(1)\ h(0)\ \ldots\ h(-N+2)]$.

Another simpler way to generate these masks is to reverse the order of the decomposition mask in both the horizontal and vertical directions. We show the comparison of the two masks in Figures 33 and 34 for the G_vH_h case only, all other reconstruction masks can be similarly related.

$g_h(-2)h_v(-2)$	$g_h(-1)h_v(-2)$	$g_h(0)h_v(-2)$	$g_h(1)h_v(-2)$
$g_h(-2)h_v(-1)$	$g_h(-1)h_v(-1)$	$g_h(0)h_v(-1)$	$g_h(1)h_v(-1)$
$g_h(-2)h_v(0)$	$g_h(-1)h_v(0)$	$g_h(0)h_v(0)$	$g_h(1)h_v(0)$
$g_h(0)h_v(1)$	$g_h(-1)h_v(1)$	$g_h(0)h_v(1)$	$g_h(1)h_v(1)$

Figure 33. Decomposition Mask G_hH_v

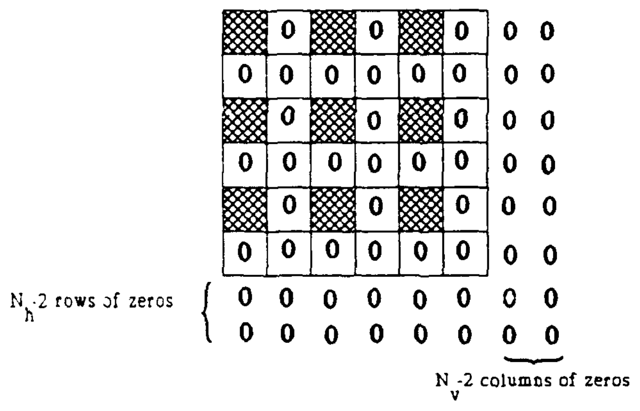
$g_h(1)h_v(1)$	$g_h(0)h_v(1)$	$g_h(-1)h_v(1)$	$g_h(-2)h_v(1)$
$g_h(1)h_v(0)$	$g_h(0)h_v(0)$	$g_h(-1)h_v(0)$	$g_h(-2)h_v(0)$
$g_h(1)h_v(-1)$	$g_h(0)h_v(-1)$	$g_h(-1)h_v(-1)$	$g_h(-2)h_v(-1)$
$g_h(1)h_v(-2)$	$g_h(0)h_v(-2)$	$g_h(-1)h_v(-2)$	$g_h(-2)h_v(-2)$

Figure 34. Reconstruction Mask H_vG_h

2. Zero-padding of the Coefficient Arrays

In order to use the reconstruction masks, we generate the work array for the lower level coefficients to accommodate for the decomposed coefficient array and mask size, and the decomposition phase that was selected.

The phase-00 reconstruction work array is the smallest of the four work spaces. The coefficient array is padded initially with one zero to the right and bottom of each of the individual coefficients. This makes the modified array have an even number of rows and columns. Then append N_h-2 columns of zeros to the right of the array followed by N_v-2 rows of zeros on the bottom.



▨ - Initial Array Coefficients in a packed format

$\boxed{0}$ - Padded zero put to the right, bottom or the bottom right of the individual packed coefficients

Figure 35. Phase-00 Reconstruction Work Array

For the phase-10 case, add one extra column of zeros to the left of the phase-00 work array. Alternatively for the phase-01 case, append one extra row of zeros to the top of the phase-00 work space. Lastly for the phase-11 situation, insert both one row and one column of zeros to the top and left of the phase-00 case.

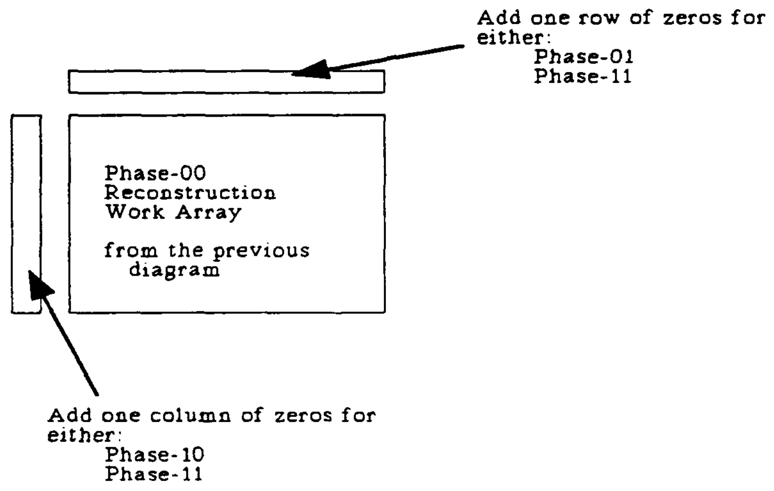

Add one row of zeros for either: Phase-01 Phase-11
Phase-00 Reconstruction Work Array
from the previous diagram
Add one column of zeros for either: Phase-10 Phase-11

Figure 36. Work Space for Phase-10,10,11 Cases

3. Two Dimensional Example

Consider the diagram of Figure 30, the following plots are the resulting mesh display of the decomposed coefficients three resolution levels down (m=-3) using a Haar wavelet and a constant phase-00 selection for each level. The top left diagram is the approximation coefficients, c_m. Going in a clockwise manner, the following plots equate to ^1d_m, ^2d_m, and ^3d_m coefficients.

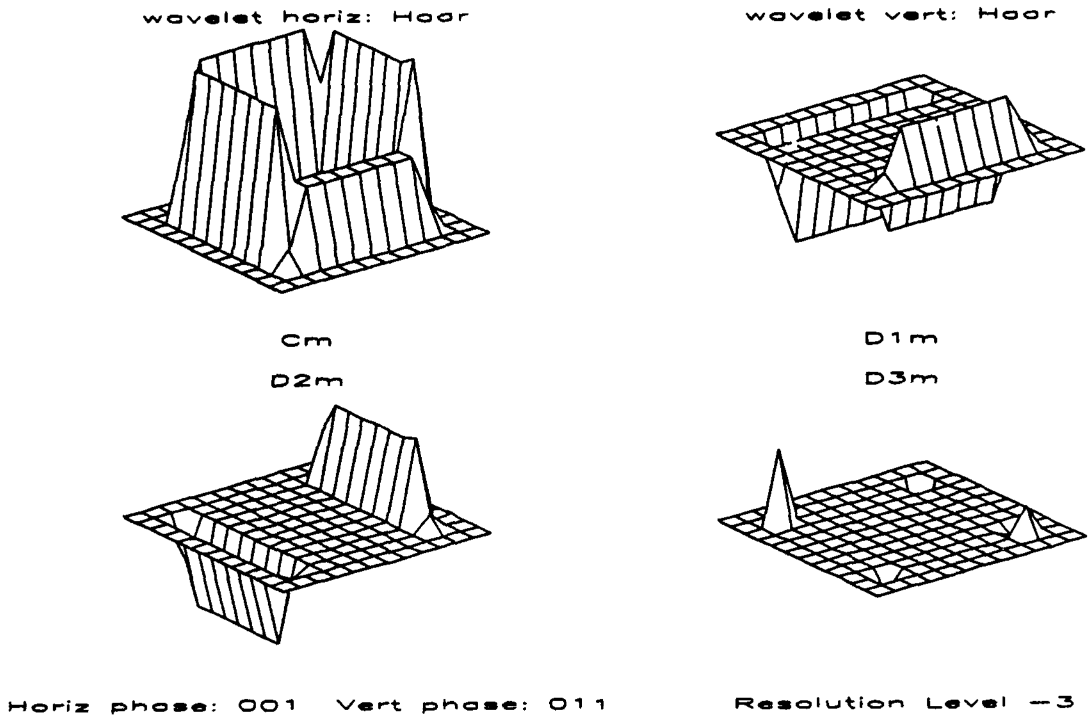


Figure 37. Mesh Display of the Coefficients for level m=-3

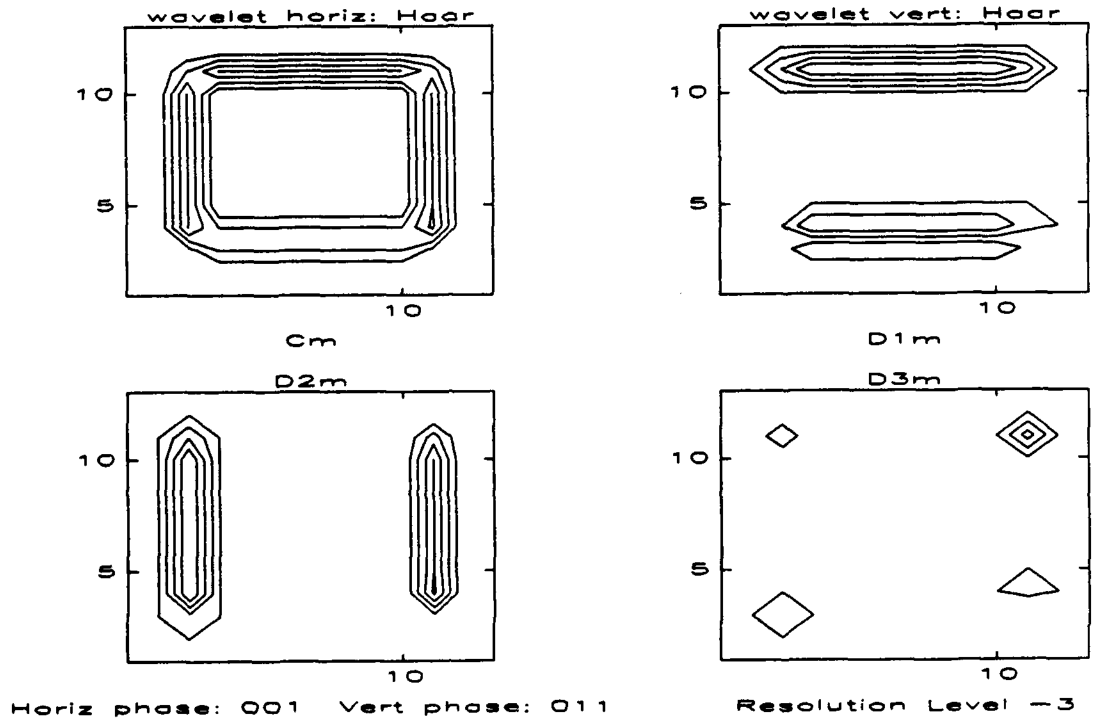


Figure 38. Contour Display of the Coefficients for level m=-3

Figures 39 and 40 compare the original and reconstructed image coefficients displayed as mesh and contour plots.

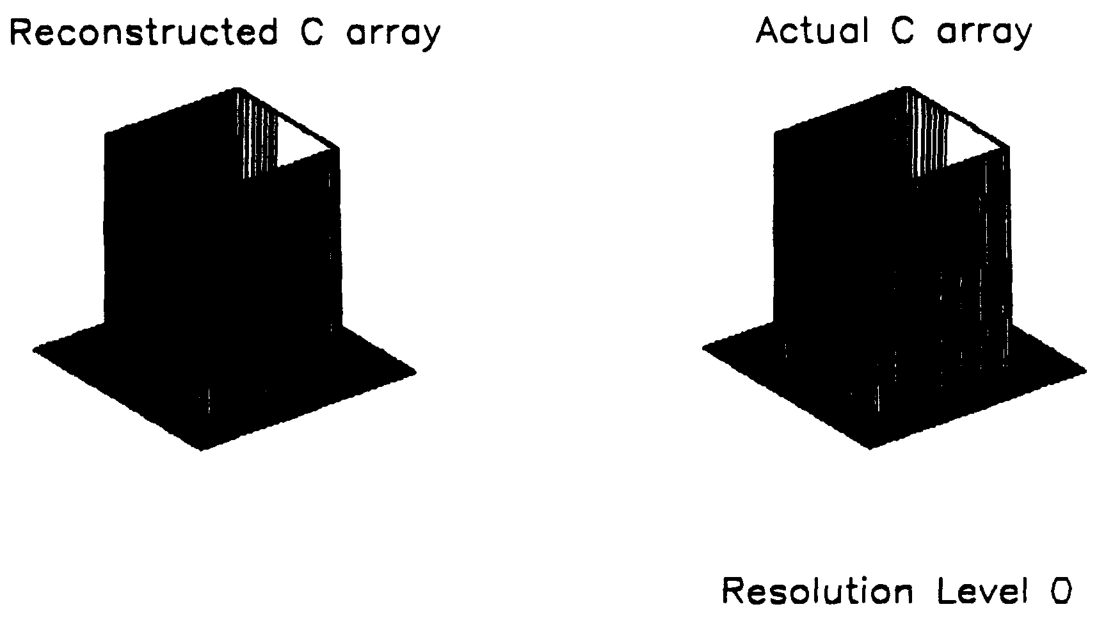


Figure 39. Reconstructed and Original Mesh Comparisons

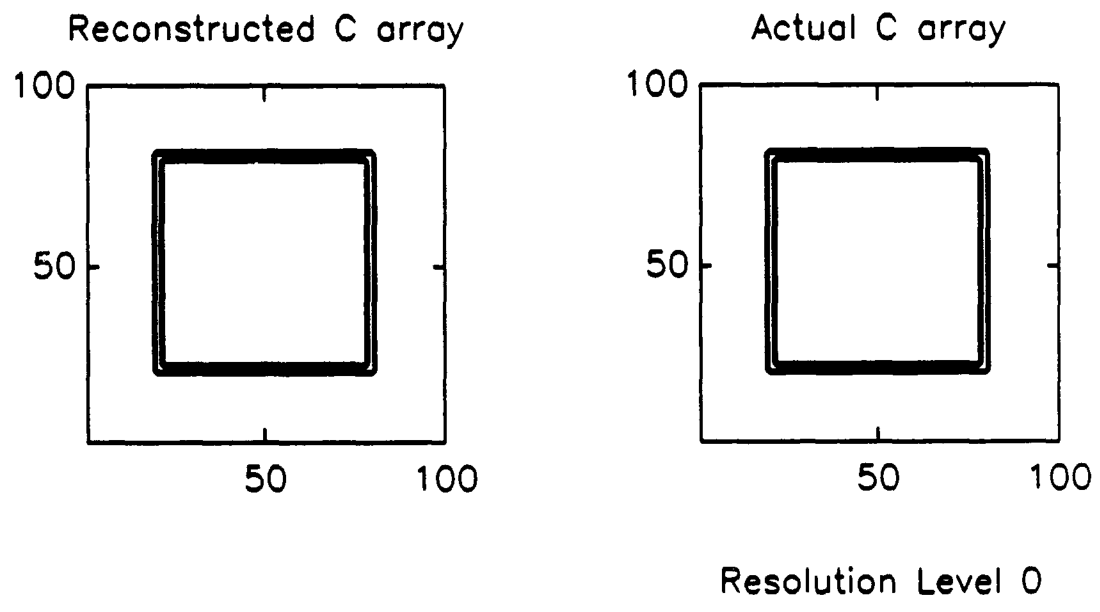


Figure 40. Reconstructed and Original Contour Comparisons

V. MULTIPLE PHASE DWT

For both the one and two-dimensional transforms, we selected a single phase for decomposing to the next lower resolution level. If we decompose down to a fixed level m=M while picking different decomposition phases for each level, we generate a different set of coefficients. This set of decomposition coefficients and any other set of decomposition coefficients will successfully reconstruct the original input array as long as we note the phase used to obtain each particular resolution level from the previous one. Such a disparity in the sets of coefficients suggests that the DWT process of a two times decimation is not shift-invariant. In other words, if we decomposed the input array with one type of decomposition scheme, our coefficients would be different if the input array is shifted before the low pass and band pass filtering processes occurs. As a note, using the energy check routine, we can see the distribution of the energy of the coefficients change as the phase decomposition scheme varies. So this shift variant property can be further exploited by varying the phase for possibly optimizing or concentrating the energy of families of signals to certain resolution levels.

However, we desire linear shift invariant (LSI) systems for the analysis of data. For the continuous WT, this property is generally true, however for the DWT case, this aspect is not. One way to bypass this obstacle is to account for all the phases during each decomposition. We call this technique the multiple-phase DWT (MP/DWT). A thorough discussion for the one-dimensional case will first be presented, which will then be extended into the two-dimensional case.

A. ONE-DIMENSIONAL MP/DWT

Consider the case of going from the data input resolution $m=0$ to $m=-1$. The coefficients that can be generated are one of two sets, depending on whether we selected the phase-0 or the phase-1 case. We really need only one of the two sets to decompose to the next lower resolution level $m=-2$ to maintain perfect reconstruction. Also, we need to pick which phase to decompose from $m=-1$ to $m=-2$. Thus for level $m=-2$, there are four possible phase combinations that can result: 00, 10, 01, and 11. The ordering of the subsequent phases is read from left to right to correspond to decomposing from level $m=0$ to $m=-2$ and we call this a phase vector. Thus ${}_{[10]}c_{-2,5}$ would be interpreted as the set of approximation coefficients at resolution level minus two and decomposed with a phase vector of [10] with an index of five. In general, for any resolution level $m \leq 0$, the total number of phase combinations is 2^{-m}.

Looking again at going from the $m=0$ to $m=-1$ case only, we can see that by sliding the filter coefficients by one instead of two units to the right, the resulting coefficients alternate on the phase selection starting with the first coefficient in the phase-0 set followed by the first coefficient in the phase-1 set as shown on the next diagram. We determine the total number of coefficients in level $m=-1$ just like a linear discrete convolution, which is the number of data points from level $m=0$ plus the length of the filter mask minus one ($| \{c_0\} | + N - 1$).

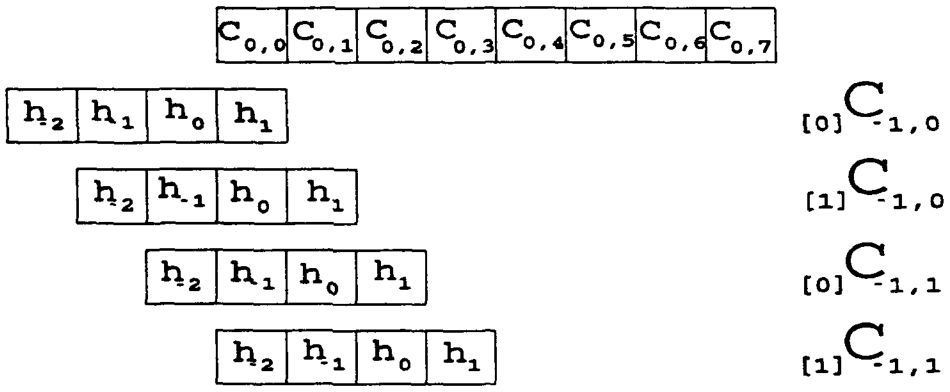


Figure 41. Alternating of Coefficients by Phase

Going from level m=-1 to m=-2 will require a slight modification before shifting the filter mask by one to the right as discussed in the previous paragraph. Since two sets of decomposition coefficients are interleaved, a zero must be appended between each coefficient in the filter mask for properly decomposing each set of coefficients. The first four values in the resulting set correspond to the phases: 00, 10, 01 and 11. This association repeats for each four values in level m=-2. In fact, the spacing of each coefficient in a desired phase is properly indexed by the same amount as the number of zeros padded for display purposes in the Chapter III.A.5.

This observation can be generalized to any arbitrary level m. In this case, separate the filter coefficients by 2^{-m}-1 zeros before the conducting single shift to the right. By noting the resolution level m, the order of the phases can be determined as the binary bit reversal of the binary values counting from zero to 2^{-m}-1. We show this characteristic as a phase-tree diagram for resolution levels m=0 to m=-3.

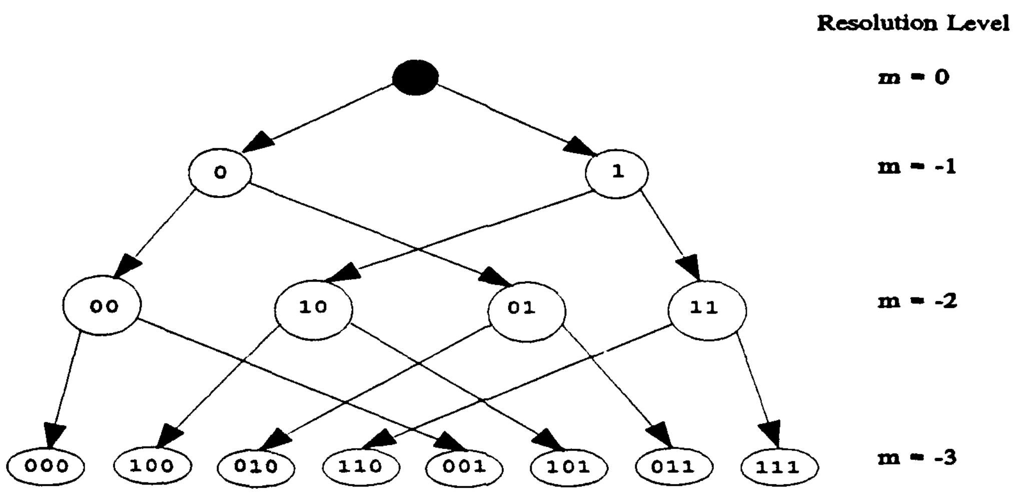


Figure 42. Phase-Tree Diagram for Resolution Levels m=0 to m=-3

We now define new filter sequences, h_m and g_m ($m \leq 0$) for decomposing the values c_{mn} to the next level $c_{m-1,n}$. The revised filter sequences are defined as

$$\begin{aligned} h_m &= [\ h(-N+2)\ \{0\}_m\ h(-N+1)\ \{0\}_m\ \ldots\ \{0\}_m\ h(0)\ \{0\}_m\ h(1)\ ] \\ g_m &= [\ g(-N+2)\ \{0\}_m\ g(-N+1)\ \{0\}_m\ \ldots\ \{0\}_m\ g(0)\ \{0\}_m\ g(1)\ ] \end{aligned} \tag{40}$$

where $\{0\}_m = 2^{-m}-1$ length zeros vector

Thus, the MP/DWT equations are given as follows [Ref. 3: p. 3][Ref. 4]:

$$\hat{c}_{m-1,i} = \sum_{k=-N+2-(N-1)(2^{-m}-1)}^{1} \sqrt{2}h_m(k)\,\hat{c}_{m,i+k-1} \tag{41}$$

and

$$\hat{d}_{m-1,i} = \sum_{k=-N+2-(N-1)(2^{-m}-1)}^{1} \sqrt{2}g_m(k)\,\hat{d}_{m,i+k-1} \tag{42}$$

The total number of coefficients also can be generalized in terms of the size of the input array, $|\mathbf{d}_0|$, the size of the h or g mirror filters, N, and the current resolution level, m:

$$|\hat{c}_m| = |c_0| + (N-1)(2^{-m}-1) \tag{43}$$

So enough memory must be allocated if we desire lower and lower resolutions, since the number of possible phases increases by a power of two each step downward.

Consider a BPSK signal shown in Figure 43. Using a Haar wavelet and a phase vector of [11111111], a contour display of the energy in the approximation coefficients and of the detail coefficients results and are shown next. Notice that much of the 180° phase shifts are not readily detectable.

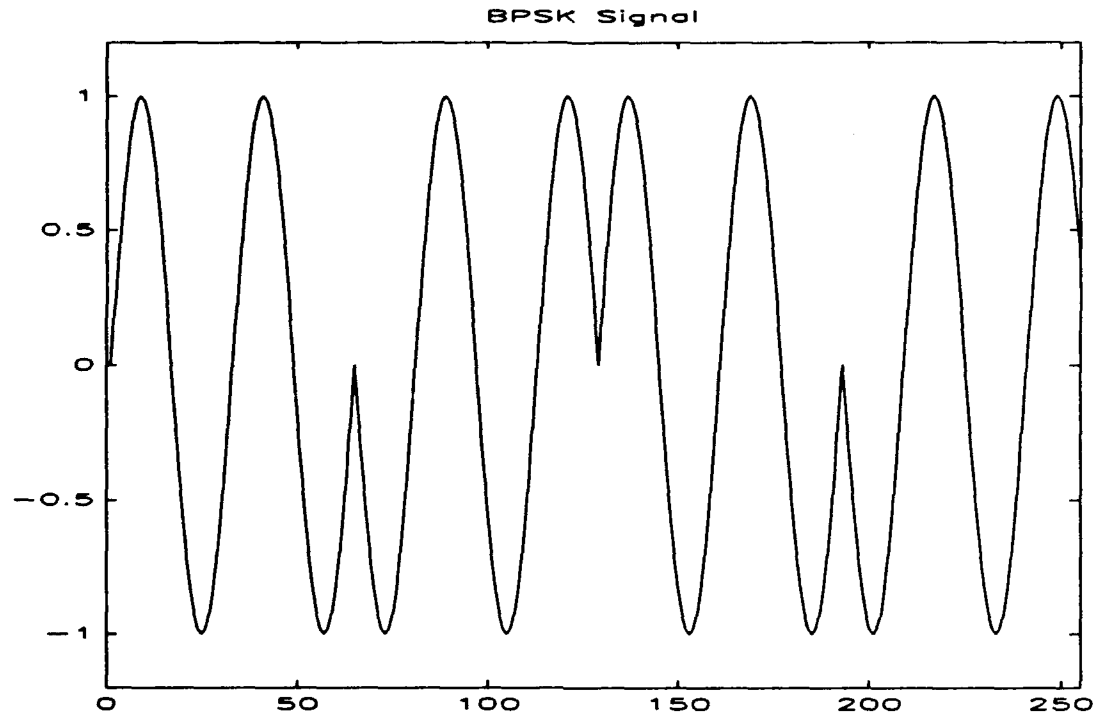


Figure 43. Sample BPSK Signal [Ref. 4]

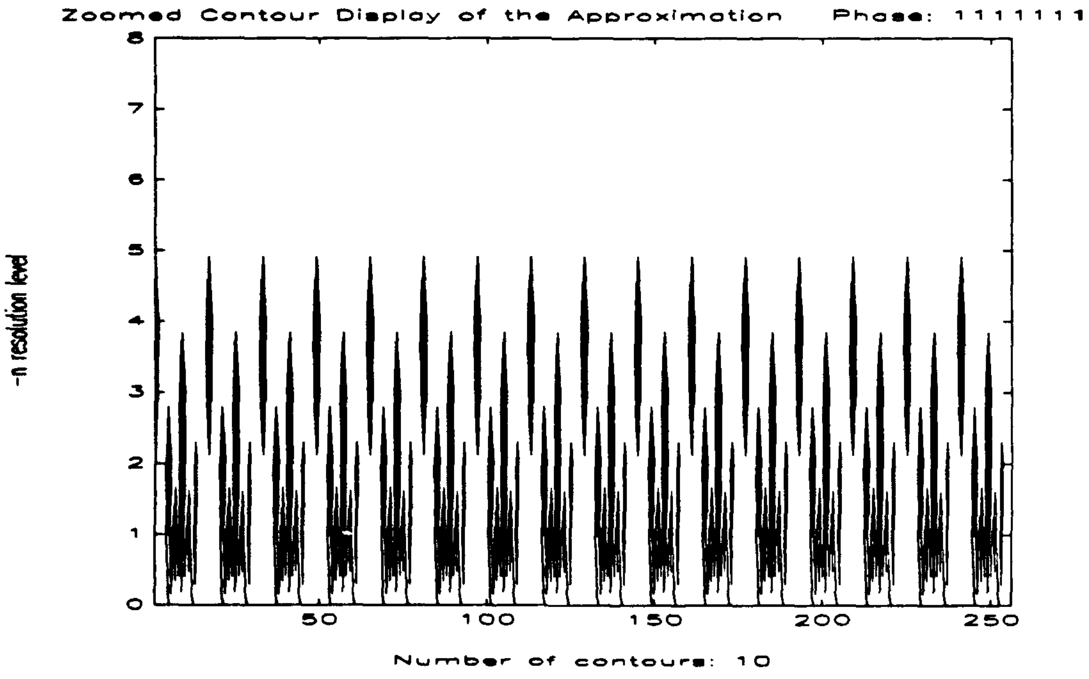


Figure 44. Approximation $|c_{nm}|^2$

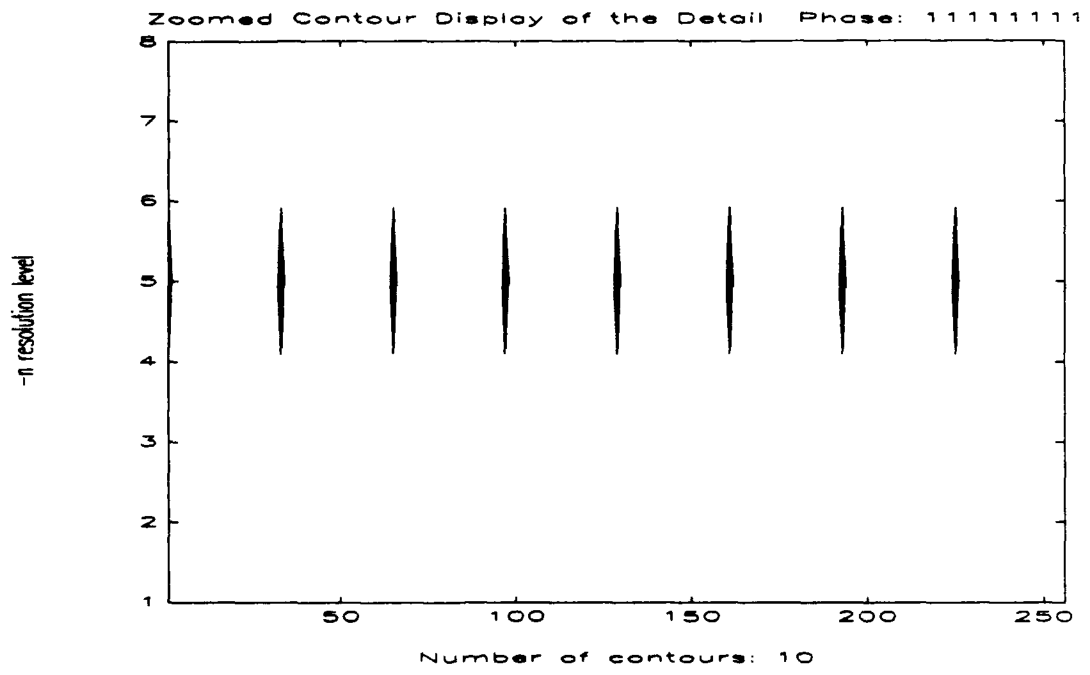


Figure 45. Detail $|d_{mn}|^2$

Now for the MP/DWT case, the 180° phase shifts of the BPSK signal are now very apparent in both the approximation and detail coefficients:

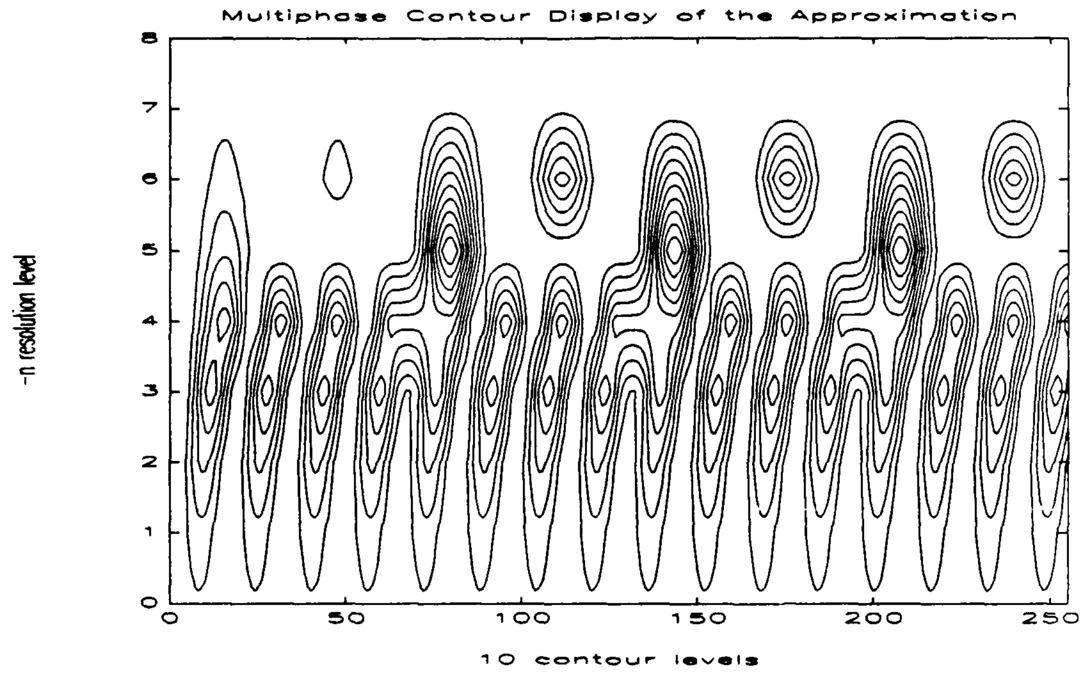


Figure 46. MP/DWT $|c_{mn}|^2$

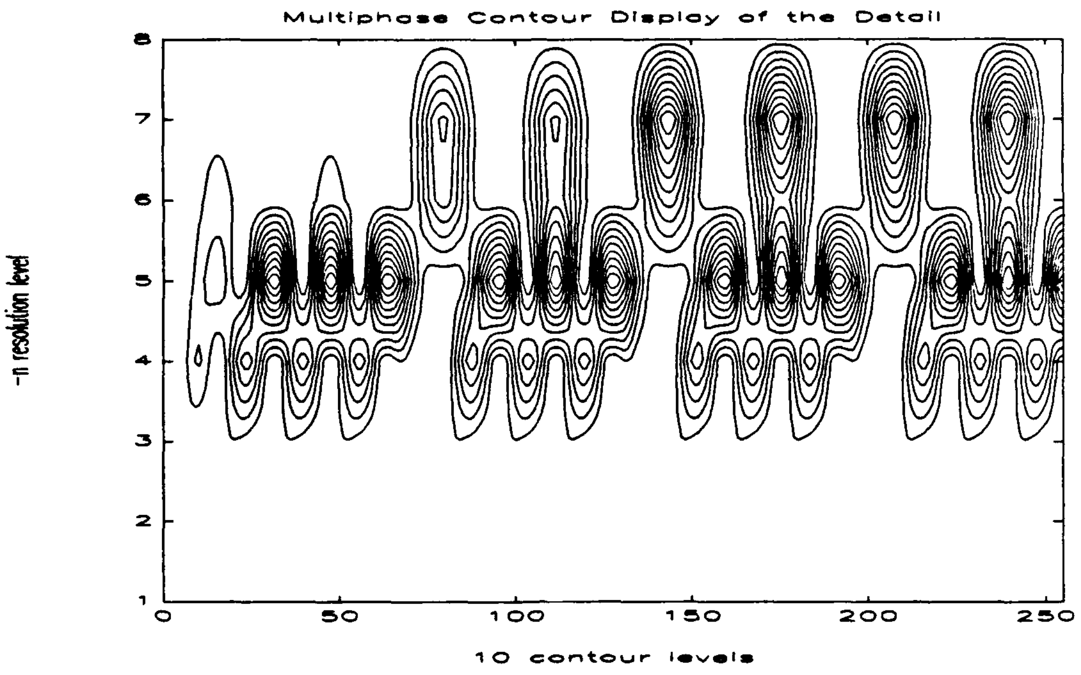


Figure 47. MP/DWT $|d_{mn}|^2$

Twice the energy of the coefficients for the current resolution level will equal the total energy on adjacent lower level's approximation and detail coefficients, since there are two valid sets of coefficients for the reconstruction to the higher resolution. So the energy checking routine discussed in the one-dimensional DWT is still valid as long as we account for a factor of one half when going up one resolution level.

The actual zero padding of the filter masks is not the most optimum method in the determination of the decomposition coefficients' proper phase order. A much more efficient technique can be determined by noting the pattern that develops as Equations 42 and 43 and writing it out explicitly. The following diagram shows the developing pattern for m=-2.

```
C_m-1        C_m                                           m = -2

c(0)         =a3c(0)   0         0          0  \           L = |c_0|
c(1)         =a3c(1)   0         0          0  |
c(2)         =a3c(2)+a2c(0)      0          0  |           N = |h| = 4
c(3)         =a3c(3)+a2c(1)      0          0  > (L-1) 2^-m
c(4)         =a3c(4)+a2c(2)+a1c(0)          0  |
c(5)         =a3c(5)+a2c(3)+a1c(1)          0  /
c(6)         =a3c(6)+a2c(4)+a1c(2)+a0c(0)                  where:
c(7)         =a3c(7)+a2c(5)+a1c(3)+a0c(1)
                                                           [a0 a1 a2 a3 ... aN-1]
                                                                   =
c(L+N-3)     =a3c(L-1)+a2c(L-3)+a1c(L-5)+a0c(L-7)          [h(-N+2) ... h(0) h(1)]
c(L+N-2)     =   0    a2c(L-2)+a1c(L-4)+a0c(L-6)                   or
c(L+N-1)     =   0    a2c(L-1)+a1c(L-3)+a0c(L-5)           [g(-N+2) ... g(0) g(1)]
c(L+N)       =   0         0    a1c(L-2)+a0c(L-4)
c(L+2N-3)    =   0         0    a1c(L-1)+a0c(L-3)
c(L+2N-2)    =   0         0          0    a0c(L-2)
c(L+2N-1)    =   0         0          0    a0c(L-1)
                 \_____________________/
                         (N-1)
```

Figure 48. MP/DWT Pattern Development for m=-2

Notice that the coefficient set, c_{m-1}, is a summation of N vectors with $|c_0| + (N-1)(2^{-m}-1)$ elements. We pad some of the vectors with zeros either in the beginning or the ending of the approximation sequence of the current level and multiply by an appropriate scalar value from one of the filter coefficients. The following equations summarizes the development:

$$\{\hat{c}_{m-1}\} = \sum_{p=0}^{N-1} a_p \, \Upsilon_m \qquad a_p = \sqrt{2}h(-N+2+p) \tag{44}$$

$$\{\hat{d}_{m-1}\} = \sum_{p=0}^{N-1} b_p \, \Upsilon_m \qquad b_p = \sqrt{2}g(-N+2+p) \tag{45}$$

where

$$\underline{Y}_m \doteq \left[\left\{ \underline{0}_{mb} \right\} \left\{ \hat{c}_m \right\} \left\{ \underline{0}_{me} \right\} \right] \tag{46}$$

$\{0_{mb}\} = \{$zeros coefficient row vector of size $(N-1-p)*2^{-m}\}$
$\{0_{me}\} = \{$zeros coefficient row vector of size $p*2^{-m}\}$.

This vectorized technique is faster than the iterative process of padding the filter coefficients appropriately. However, we need more buffer space to calculate the data, which is dependent on the number of resolution levels desired.

The reconstruction process works in a very similar fashion as in the DWT case. All we need is the selection of the coefficient sets, which in turn determines the unique path back upwards in the phase-tree diagram, similar to Figure 8. Once we select the desired phase at the lowest resolution level, the we extract the corresponding approximation and detail coefficients and put them in a packed format. Then the DWT reconstruction commences to get up one level, where we extract the next set of detail coefficients. We repeat this process until we reach resolution level m=0.

B. TWO-DIMENSIONAL MP/DWT

Since we define the basis functions as separable in the two orthogonal directions, we can use the same arguments discussed for the one-dimensional case. When we talk about coefficients and masks, they are now two-dimensional arrays as discussed in Chapter IV. Thus, the basic equations for the two-dimensional MP/DWT are

$$\hat{c}_{m-1}(a,b) = 2 \sum\sum_{\substack{j=-N_h+2-(N_h-1)(2^{-m}-1) \\ k=-N_v+2-(N_v-1)(2^{-m}-1)}}^{1} H_hH_v(j,k)\hat{c}_m(a+j-1,b+k-1) \tag{47}$$

$$^1\hat{d}_{m-1}(a,b) = 2 \sum\sum_{\substack{j=-N_h+2-(N_h-1)(2^{-m}-1) \\ k=-N_v+2-(N_v-1)(2^{-m}-1)}}^{1} H_hG_v(j,k)\hat{c}_m(a+j-1,b+k-1) \tag{48}$$

$$^2\hat{d}_{m-1}(a,b) = 2 \sum\sum_{\substack{j=-N_h+2-(N_h-1)(2^{-m}-1) \\ k=-N_v+2-(N_v-1)(2^{-m}-1)}}^{1} G_hH_v(j,k)\hat{c}_m(a+j-1,b+k-1) \tag{49}$$

$$^3\hat{d}_{m-1}(a,b) = 2 \sum\sum_{\substack{j=-N_h+2-(N_h-1)(2^{-m}-1) \\ k=-N_v+2-(N_v-1)(2^{-m}-1)}}^{1} G_hG_v(j,k)\hat{c}_m(a+j-1,b+k-1) \tag{50}$$

where the top left corner of the mask corresponds to location (a,b)=(0,0), and the positive directions are to the right and downward [Ref. 3: pp. 2-3]:.

Even the one-dimensional vectorized process converts into a "matricized" process. Here, the resulting coefficient array is the matrix addition of N_hxN_v total matrices, each appropriately padded with zeros on all four sides and multiplied by one of N_hxN_v scalar values. The two-dimensional equations result as follows:

$$\{\hat{c}_{m-1}\} = \sum_{p=0}^{N_h-1} \sum_{q=0}^{N_v-1} a_{pq}\, Z_m \qquad a_{pq} = 2H_hH_v(-N+2+p,-N+2+q) \tag{51}$$

$$\left\{{}^{1}\hat{d}_{m-1}\right\} = \sum_{p=0}^{N_h-1}\sum_{q=0}^{N_v-1} b_{pq}\, Z_m \qquad b_{pq} = 2H_hG_v(-N+2+p,-N+2+q) \tag{52}$$

$$\left\{{}^{2}\hat{d}_{m-1}\right\} = \sum_{p=0}^{N_h-1}\sum_{q=0}^{N_v-1} c_{pq}\, Z_m \qquad c_{pq} = 2G_hH_v(-N+2+p,-N+2+q) \tag{53}$$

$$\left\{{}^{3}\hat{d}_{m-1}\right\} = \sum_{p=0}^{N_h-1}\sum_{q=0}^{N_v-1} f_{pq}\, Z_m \qquad f_{pq} = 2G_hG_v(-N+2+p,-N+2+q) \tag{54}$$

This time we define Z_m graphically in the following figure:

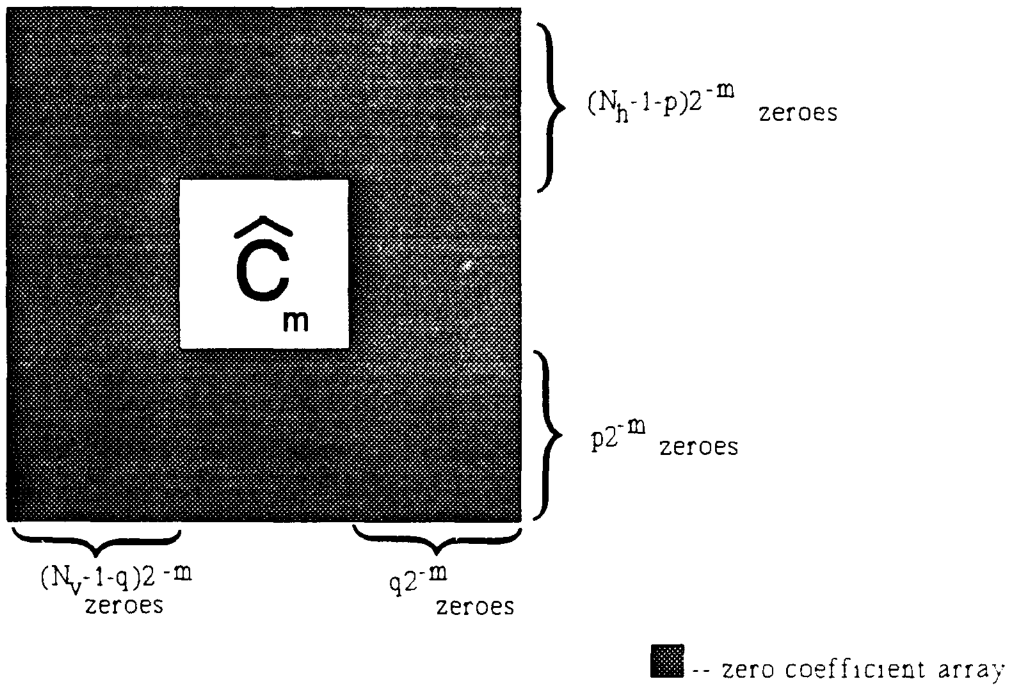


Figure 49. Definition of Z_m

The phase-tree diagram can be viewed in three dimensions to more appropriately show the proper interleaving of the phases. Since there are four possible phases, there will be four more locations in the indexing of the data on the next lower resolution level. The total number of coefficients would be

$$[Rows_0+(N_v-1)(2^{-m}-1]\cdot[Cols_0+(N_h-1)(2^{-m}-1)] \tag{55}$$

where Rows_0 and Cols_0 are the number of row and columns of the original image. In order to view the phase-tree diagram correctly, we must consider each resolution level as a new plane for the ordering of the coefficients.

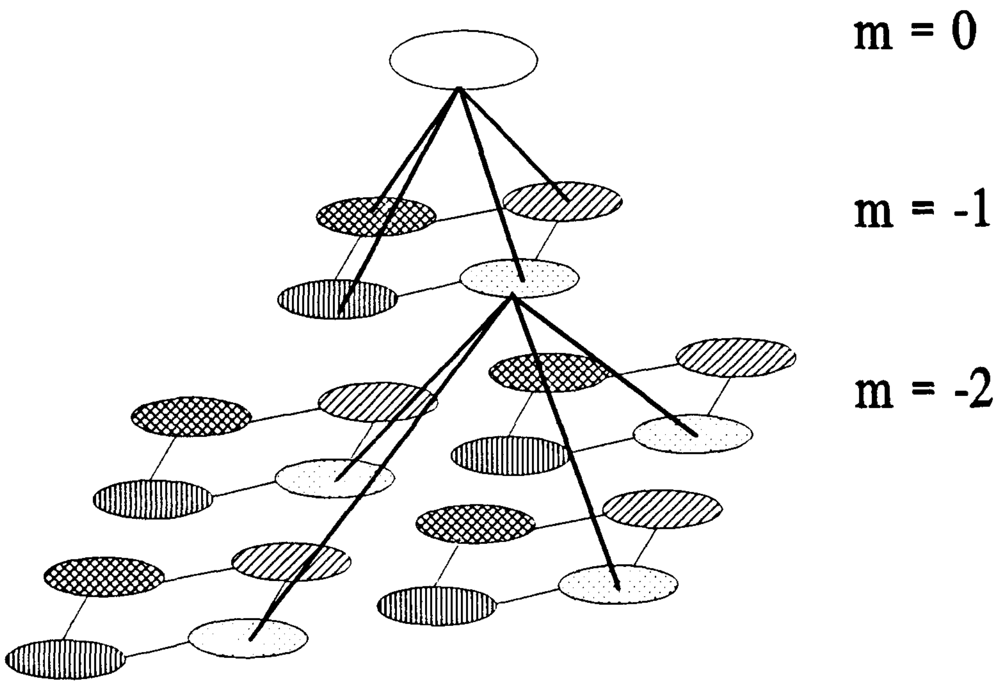


Figure 50. Phase-Tree Diagram

The MP energy display of the coefficients closely resembles the display for the single phase case. This energy display differs by the fact that they are the normalized average energy values for each level. As before, we normalize the energy to the resolution level m=0. Consider as an example, a MP/DWT of the image in Chapter IV using Haar wavelets and decomposing down to m=-3. The following display shows the MP/DWT energy plot:

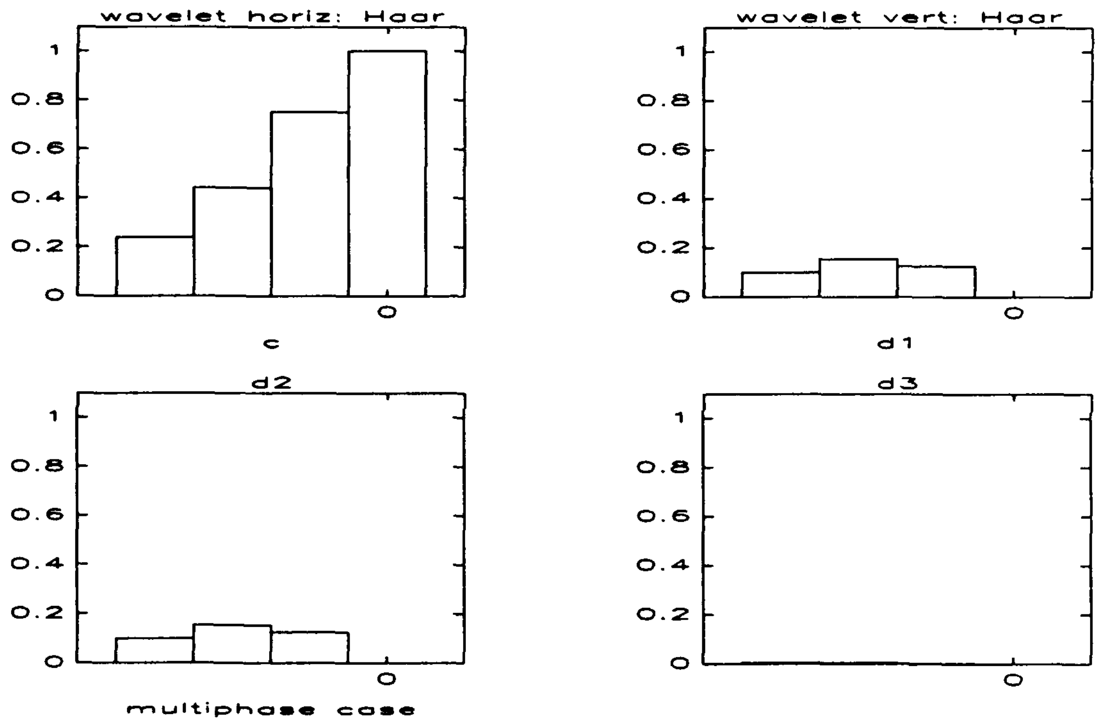


Figure 51. MP/DWT Energy Display

and the energy check is as follows:

```
          Verification of the energy in each resolution level

 Level      Energy in             Energy in C          Difference
   m        level m-1=              level m
            c+d1+d2+d3
 ----------------------------------------------------------------
 -2     0.441935911016949     0.441935911016949     0.000000000000000
 -1     0.751059322033898     0.751059322033898     0.000000000000000
  0     1.000000000000000     1.000000000000000     0.000000000000000
```

The MP/DWT of the three-dimensional energy display for all the coefficients of the image and the corresponding contour images for the m=-3 case are as follows:

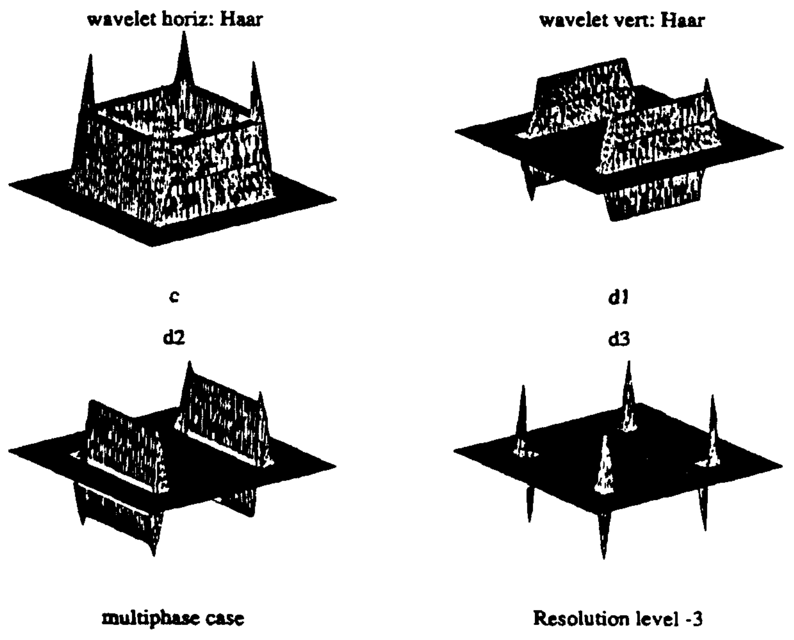


Figure 52. Mesh Display of the Stored Energy for m=-3

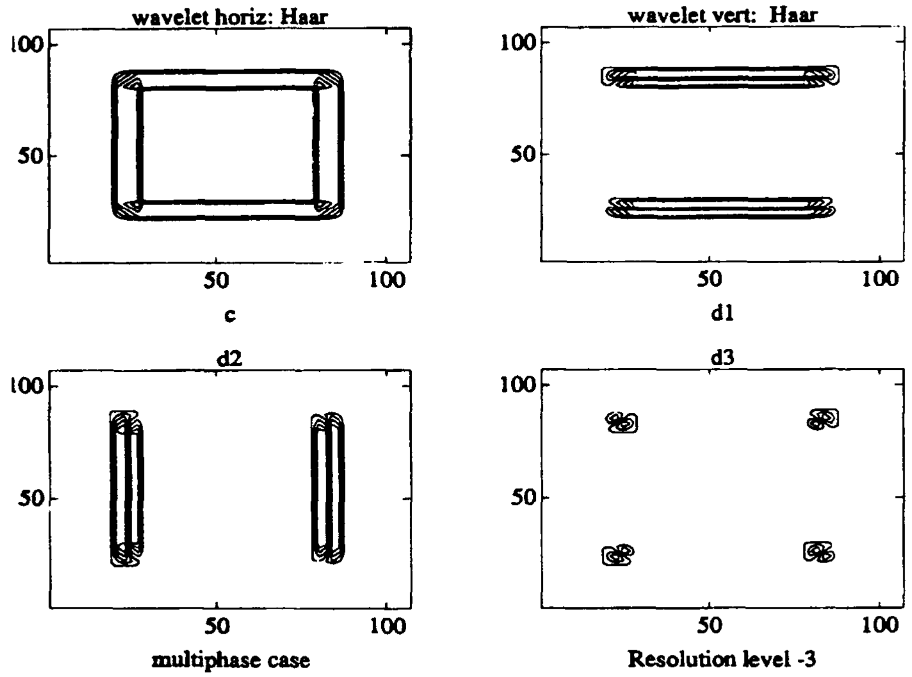


Figure 53. Contour Display of the Stored Energy for m=-3

VI. MATLAB DWT ROUTINE DESCRIPTION

The Appendix lists the developed DWT algorithms. We categorize them primarily into two classifications, the one and two-dimensional cases. In each situation, we further define two subgroups, the single phase DWT and the MP/DWT. Included with the software is an ASCII version of this chapter in a README.TXT file. This chapter first describes the system configuration to run the routines. Then we list brief steps for each of the two general cases. These steps are not totally inclusive, but the program's prompts will fill any other gaps in the routine. Finally, we discuss important considerations for obtaining graphical output of the results since this is highly dependent on the system hardware.

A. SYSTEM CONFIGURATION

Minimum hardware requirements for the PC version are an Intel-386 microprocessor with a math co-processor, Microsoft DOS or Digital Research DOS 5.0, VGA, at least three megabytes of hard disk space, and four megabytes of RAM, for most of the DWT applications. However, we recommend at least eight megabytes of RAM for using arrays greater than 10,000 elements, or if $m \leq -5$ in the two-dimensional MP/DWT case. Also, some DOS memory managers may conflict with the Pharlap DOS extenders that MATLAB Version 3.5 uses. For the Sun workstations, in order to operate PROMATLAB Version 3.5, we recommend the Open Windows or Sunview graphical environments with at least eight megabytes of RAM. Any further memory can be negotiated with the system administrator. Note that UNIX-based operating systems discriminate characters between upper and lower case letters.

B. GENERAL PROCEDURAL STEPS

Initially, we must be in the MATLAB environment prior to running any of the routines. Either generate in MATLAB or load the sampled data for resolution level $m=0$, and assign a variable name. If and files with the prefix, "leg", and the suffix, ".met" (all in lower-case letters), exists in the current directory, they will be deleted once the DWT routines commence. Thus if you desire to retain these files, rename them. The DWT routines use these files as the output graphic files for the routines.

1. One-Dimensional General Procedures

a. If you have no data, the routines include three files as data that can be loaded. Type "load" followed by a space and then one of the three names: "et60.m", "et90.m", or "bpsk". The variable name will be either "et60", "et90", "bpsk".

b. Invoke "wav1d" in lower-case letters.

c. Enter the variable name of the input data.

d. Enter the sampling frequency, <RET> if not known.

e. Select the desired h coefficients.

f. Select "Y" if you desire the MP/DWT routine and skip to step k.

g. In you did not select the MP/DWT routine, the single phase DWT commences, decomposing the coefficients until resolution level $m=-\lceil \log_2(| c_n |) \rceil$ with an initial query of if you desire to see the intermediate resolution level displays.

h. Choose either the phase-0 or phase-1 decomposition for each resolution level.

i. After the final decomposition, the energy checks commence.

j. Select "Y" if you desire to do the reconstruction process. If so, answer yes or no if you want to display the intermediate levels.

k. Lastly, the time/scale display commences, and the series of questions ask if you want to zoom in or out and what number of contour levels you desire.

l. For the MP/DWT case, steps i to k are essentially the same except for the reconstruction, where you must initially select the desired phase, thereby fixing the phase path back up to level m=0.

2. Two-Dimensional Procedures

a. If you do not have any input data, invoke "idata" for a small selection of images. The variable "im" (in lower-case letters) will be the input for step c.

b. Invoke "wav2d" in lower-case letters.

c. Enter the variable name of the input data array.

d. Select the desired h coefficients in the horizontal direction.

e. Select the desired h coefficients in the vertical direction.

f. Select "Y" if you desire the MP/DWT routine.

g. Enter the number of resolution levels to go down. An initial good selection is four since this will cover all four possible phases.

h. For the single phase DWT case, select one of four of the decomposition phases for each level.

i. After the final decomposition, the energy checks commence.

j. For the single phase DWT case, select "Y" if you desire the reconstruction routine.

C. OBTAINING GRAPHICAL OUTPUT

Except for the initial time/frequency plots, all other displays will query you if you want to store the plot. If so, the plot is stored as a metafile with the prefix of "leg" and the suffix ".met". A number starting from zero is put between the prefix and suffix, for example, leg14.met would correspond to the fifteenth stored plot in the DWT routine. Output of these metafiles are hardware and system dependent. Refer to the GPP command in the MATLAB User's Guide for more detailed information. Two C-shell script files, called "meta13" or "meta14", will generate a temporary Postscript file for plotting on a Postscript printer all the metafiles retained from the DWT routine. The use of these files is currently only guaranteed to work at the US Naval Postgraduate School Electrical Engineering Department's Sun workstations, and we list them here as a guide in generating other C-shell script files for a particular system. For the DOS version, the following command will generate all the metafiles onto a HP Laserjet III printer as long as the we invoke the following DOS command in the same directory as the metafiles: **for** %f **in** (leg*.met) **do** gpp386 %f /djet150 /ol /fprn . If you put it in a DOS batch file, echo the percent sign (%% instead of %). Please note that the computer may take some time generating these plots. Be also aware that depending on the size of the plot, the graphic files may exceed the printer buffer memory.

VII. CONCLUSIONS

In this thesis, we developed discrete wavelet transform algorithms for both the one and two-dimensional cases. The iterative mechanism of decomposing the data coefficients by a convolution-and-subsample process is achievable. These realizations require that the scaling and wavelet functions form orthonormal sets with compact support. Also for the two-dimensional case, the basis functions were separable in the two orthogonal directions.

We found that the output of the decomposition coefficients can be causal provided that the definitions of the indices for both filter coefficients, h and g, must be selected to accommodate the causal condition. In fact, the definition of the g filter coefficients must be adjusted from the definitions given by Mallat and Daubechies. We found for the one-dimensional case, the decomposition is simply a dot product of the filter coefficients with a set of values windowed from the higher resolution level approximation coefficients. This window subsequently shifts two units to the right for the next decomposed coefficient. We found that the reconstruction process uses the same decomposition algorithm if we separate each of the individual input values by a zero, the filter coefficients are time-reversed, and the shift is now one unit to the right at a time.

The two-dimensional case was a natural extension of the one-dimensional process. We found that the decomposition can be realized as a mask element multiplication and summation, with the mask shifting two units to the right until the row completion and then two units down to process the next row of coefficients. The shifting of the mask is similar to a raster scan. The reconstruction follows the same process as in the decomposition but each of the input array coefficients are separated by a row and a column of zeros, the filter masks are spatially reversed in both direction, and the raster shift is one unit instead of two.

We found that there were two possible phases in decomposing the data in the one-dimensional case, and four possible phases for the two-dimensional case. This led to the conclusion the current decomposition algorithm was very shift variant, contrary to the continuous version to the wavelet transform, which is shift invariant. We then determined that the discrete case can possibly approximate the continuous version if we account for all the phases during the decomposition. We then developed the multiple-phase discrete wavelet transform for both one and two dimensions. The coefficient sets for a particular phase can be interleaved with other sets of a different phase. We found that the multiple-phase was more capable in detecting discontinuous properties of data than in the regular discrete wavelet transform, such as a binary phase shifting function.

APPENDIX DWT MATLAB FILE LISTINGS

A. ONE-DIMENSIONAL ROUTINES

```
%*************************************************************
%                                          15 SEP 92
% WAV1D.M    The One Dimensional Discrete Wavelet Transform
%
%
% By:   LT  J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:  (408) 646-3044/2772
%
%*************************************************************
%
% This routine does a one-dimensional dicrete wavelet decomposition
% of a one-dimensional data array.  The algorithm allows for the
% selection of the desired phase for the respective decompositon
% level.  The following routines are necessary for the proper
% operation of this algorithm:
%
%   chkputer.m -- algorithm checks for compatible hardware
%   daubdata.m -- determines the h coefficients for a daubechies
%                 compactly supported orthogonal wavelet
%   mp.m       -- MultiPhase decomposition
%   phs1.m     -- selects the phase-1 decomposition
%   phs0.m     -- selects the phase-0 decomposition
%   dpscoef.m  -- display the coefficients for each level
%   enrg1d.m   -- determines the energy check for each resolution
%   recon1d.m  -- reconstruction algorithm for verification
%   plane.m    -- displays the coefficients in the time-scale domain
%   strplt.m   -- stores the plots in meta-files
%-----------------------------------------------------------------
clc
chkputer  % Checking for the basic hardware necessary


%-----------------------------------------------------------------
% c0 corresponds to the resolution level 0, the original data
%    array.
clc
c0=input('Enter the name of the data in vector row format: ');

% We will check if the data is in row format, if in column format,
% take the transpose of the input.

[i1 j1]=size(c0);
   if       j1==1, c0=c0';
   elseif   i1>1 & j1>1,
            disp('Bad Input Data, Restart the Program')
            return
   end
clear i1 j1
%-----------------------------------------------------------------
```

```
fsamp=input('Enter the sampling frequency (Hz):  ');


%------------------------------------------------------------------
% The following code determines the desired wavelet

for i=1:8,disp([' ']),end
q1=['Enter the number for the desired choice: '
    '                                          '
    '(1) Haar h coefficients                   '
    '(2) Daubechies h coefficients             '
    '(3) Own set of h coefficients             '
    '                                          '];
disp(q1)
qqq=input(' ');

  if     qqq==3,
         h=input('Enter your own set of h coefficients');
          %Checking if the data are compact support
         a=sum(h);b=h*h';
         toll=1e-12;
         while abs(1-a)>toll & abs(.5-b)>toll
          disp('Your h coefficients are NOT compactly supported!')
          h=input('Re-enter the h coefficients or ctrl-z to stop');
         end
         wwavlet='OTHER';
  elseif qqq==2,
         qq=input('How many taps (2-10)? ');
         h=daubdata(qq);
         wwavelet=['Daubechies ',num2str(qq),'-tap'];
  else    h=[.5 .5];wwavelet='HAAR';
  end

h=sqrt(2)*h;    % The factor of sqrt(2) normalizes the energy

%------------------------------------------------------------------
% Some definitions of variables used throughout the entire
%      algorithm:
%
% wwavelet -- name of the selected wavelet
% LL       -- the length of the input array
% Nh       -- the number of h and thus g coefficients
% lowest   -- the number of resolution levels below the input
% pltcnt   -- plot counter for storing desired plots
% h        -- the "h" coefficients
% g        -- the "g" coefficients
% lowest   -- the lowest resolution level
% phsvct   -- the phase vector for recording the appropriate phase
% zro      -- zero vector to record the appropriate zero padding
%             for each resolution level.

%------------------------------------------------------------------


%------------------------------------------------------------------
% We will intialize some of the constants:

LL=length(c0);
Nh=length(h);
lowest = -ceil(log(LL)/log(2));
lvl=0;
pltcnt=0;
%------------------------------------------------------------------
```

```
%----------- -----------------------------------------------------
% Generating the g coefficients

g=fliplr(h);
for i=2:2:Nh
  g(1,i) = -g(1,i);
end
%----------------------------------------------------------------




%----------------------------------------------------------------
% The following set of lines branches to the multiphase
% decomposition algorithm if desired. Program ends after
% the multiphase execution
clc
q=input('Do you desire the muliphase decomposition (Y/N)? ','s');
if q=='Y' | q=='y'
  mp
  return
end
%----------------------------------------------------------------




%----------------------------------------------------------------
% The Decomposition Routine

% Determine if we want to ignore the display for each resolution
%     level's coefficients.
plt1=input('Bypass the display of the Coefficients (Y/N)? ','s');

clc
phs=2;  % This just sets "phs" initially to be NOT equal to 1 or 0
phsvct=[];
zro=[];


% ----------------------------------------------------------------
% The following code sets the number of variables necessary
% to record the coefficients. ie. lowest=-2, we have
%                        c0,c1,c2,d1,d2
LLL=LL;
for lvl=-1:-1:lowest
 while 1==1
   phs=input('Enter the desired Phase [0/1] for this resolution level: ');
   if phs==1 |phs==0,break,end
 end
 phsvct=[phs phsvct];
 eval(['c',num2str(abs(lvl)),'=phs',num2str(phs),'(c',...
      num2str(abs(lvl)-1),',h);'])
 eval(['d',num2str(abs(lvl)),'=phs',num2str(phs),'(c',...
      num2str(abs(lvl)-1),',g);'])

 % This if statement runs the display of the coefficients if the flag
 %       plt1 is set to NO
 if plt1=='N' | plt1=='n', dspcoef, end
 phs=2
 eval(['LLL=max(LLL,length(c',num2str(-lvl),')*2^(-lvl));'])
 zro=[zro 2^(-lvl)-1];
end

%----------------------------------------------------------------
% We now do an energy check of the coefficients
```

```
enrg1d
clg
%-----------------------------------------------------------------

%-----------------------------------------------------------------
% We now run the reconstruction option

recon1d

%-----------------------------------------------------------------


%-----------------------------------------------------------------
% We now display the coefficients in the time-scale domain

plane

%-----------------------------------------------------------------
%      END OF WAV1D.M





%-----------------------------------------------------------------
% Phase Zero decomposition for the 1-D case
%
% By:    LT J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:  (408) 646-3044/2772
%-----------------------------------------------------------------
% phs0.m is a function m-file the does a phase 0 decomposition of
% of the data array using either the h or g coefficients. The main
% program that calls this routine is "wav1d.m"
%
% The input arguments for this routine:
%
%       data   --  input data
%       h      --  the h or g coefficients
%       L      --  the length of the data vector
%       H      --  the length of the h or g coefficient vector
%-----------------------------------------------------------------


function x=phs0(data,h)

L=length(data);
H=length(h);


% The following 4 lines check if the data array is even, for the
%      phase 0 case, we must pad it with one zero if true
if rem(L/2,floor(L/2)) == 0
   data=[data 0];
   L=L+1;
end
data=[zeros(1,H-1) data zeros(1,H-2)];
L=L+2*H-3;
a=0;
for ii=1:2:L-(H-1)
 a=a+1;
 x(1,a)=data(ii:ii+H-1)*h';
end
```

```
%------------------------------------------------------------------
% Phase One decomposition for the 1-D case
%
% By:    LT  J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:   (408) 646-3044/2772
%------------------------------------------------------------------
% phs1.m is a function m-file the does a phase 1 decomposition of
% of the data array using either the h or g coefficients.  The main
% program that calls this routine is "wav1d.m"
%
% The input arguments for this routine:
%
%     data   -- input data
%     h      -- the h or g coefficients
%     L      -- the length of the data vector
%     H      -- the length of the h or g coefficient vector
%------------------------------------------------------------------
function x=phs1(data,h)

L=length(data);
H=length(h);

% The following four lines checks if the data vector is odd, if it
%     true for the phase 1 case, we must pad it with one zero
if rem(L/2,floor(L/2)) ~= 0
  data=[data 0];
  L=L+1;
end

data=[zeros(1,H-2) data zeros(1,H-2)];
L=L+2*H-4;

a=0;
for ii=1:2:L-(H-1)
 a=a+1;
 x(1,a)=data(1,ii:ii+H-1)*h';
end




%------------------------------------------------------------------
% dspcoef.m    Display the coefficients
%
% By:    LT  J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:   (408) 646-3044/2772
%------------------------------------------------------------------
% dspcoef.m displays the coefficients for each particular
% resolution level with the appropriate scaling factor.  There is
% also a special routine embedded in this algorithm for the
% proper bar display of the HAAR case. The main program is "wav1d.m"
%
% The variables are defined as follows:
%
%     z      -- the number of zeros to be padded btwn coeff's
%     lvl    -- variable from "w.m" for resolution level
%       fsamp    -- sampling frequency of the data
%       wwavelet -- string for the selected wavelet
%-----------------------------------------------------------------------

clg
hold off
axis('normal')

% For the HAAR case do the following:
if wwavelet(1) == 'H' | wwavelet(1) == 'h'
   n=2^(-lvl);
   eval(['ctemp=2^(lvl/2)*c',num2str(-lvl),';'])
   eval(['dtemp=2^(lvl/2)*d',num2str(-lvl),';'])
   ymax=max(max(ctemp),max(dtemp));
   if ymax<0,ymax=0;end
   ymin=min(min(ctemp),min(dtemp));
   if ymin>0,ymin=0;end
   L=length(ctemp);
     if L==1
       axis([0 (n*1.5-1)/fsamp 1.2*ymin 1.2*ymax]);
        subplot(211)
        plot([0 0 n*1.5-1 n*1.5-1]/fsamp,[0ctemp ctemp 0])
        title(['Approximation at Resolution level ',num2str(lvl)])
        xlabel('Haar wavelet')
       axis([0 (n*1.5-1)/fsamp 1.2*ymin 1.2*ymax]);
        subplot(212)
        plot([0 0 n*1.5-1 n*1.5-1]/fsamp,[0dtemp dtemp 0])
        title(['Detail                Phase-',num2str(phs)])
       pause
       strplt

     else
       x=[0:n:n*L-1]+.5*n;
       x=x/fsamp;
       axis([0 x(length(x)) 1.2*ymin 1.2*ymax]);
       subplot(211),bar(x,ctemp)
       title(['Approximation at Resolution level ',num2str(lvl)])
       xlabel('Haar Wavelet')
       axis([0 x(length(x)) 1.2*ymin 1.2*ymax]);
       subplot(212),bar(x,dtemp)
       title(['Detail                Phase-',num2str(phs)])
       pause
       strplt
     end

% For NON-HAAR cases, do the following:
else
   z=2^(-lvl)-1;
   eval(['ctemp=2^(lvl/2)*ptzero(c',num2str(-lvl),',z);'])
   eval(['dtemp=2^(lvl/2)*ptzero(d',num2str(-lvl),',z);'])
   ymax=max(max(ctemp),max(dtemp));
   if ymax<0,ymax=0;end
   ymin=min(min(ctemp),min(dtemp));
   if ymin>0,ymin=0;end

    % We can display the data with a window size of the original
    %    input, or show the spillover values
    q=['Enter the type of display:'
       '                          '
       '   1 - Windowed Display   '
       '   2 - Non-Windowed       '];
   clc
   disp(q)
   disp(['  '])
   q=input(' ');
```

```
    if q==2
      x=[0:length(ctemp)-1]+.5;
      x=x/fsamp;
      axis([0 x(length(x)) ymin ymax]);
      subplot(211),bar(x,ctemp)
      title(['Approximation at Resolution level ',num2str(lvl)])
      xlabel([wwavelet,' Wavelet'])
      axis([0 x(length(x)) ymin ymax]);
      subplot(212),bar(x,dtemp)
      title(['Detail                    Phase-',num2str(phs)])
      xlabel(['Non-Windowed display  -- max windowed value: ',...
             num2str(length(x)-1)])
      pause
      strplt
    else
      x=[0:LL-1]+.5;
      x=x/fsamp;
      axis([0 x(length(x)) ymin ymax]);
      subplot(211),bar(x,ctemp(1,1:LL))
      title(['Approximation at Resolution level ',num2str(lvl)])
      xlabel([wwavelet,' Wavelet'])
      axis([0 x(length(x)) ymin ymax]);
      subplot(212),bar(x,dtemp(1,1:LL))
      title(['Detail                    Phase-',num2str(phs)])
      xlabel(['Windowed display -- max windowed value: ',num2str(LL-1)])
      pause
      strplt
    end
  end




%*****************************************************************
% enrg1d.m     Determining the Energy in each resolution level
%                          One Dimensional Case
%
% By:    LT  J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:  (408) 646-3044/2772
%*****************************************************************
% For the single phase case, the energy is normalized to resolution
% level 0.  All the energies of the "c's" and the "d's" of a
% particular level should add up to the energy of the "c's" of the
% next higher level.
%
%      enrgc -- energy array of the "c's"
%      enrgd -- energy array of the "d's"
%      norm  -- energy of the data "c0"
%      esum  -- energy sum array (enrgc+enrgd)
%
% The calling routine is "wav1d.m" .
% strplt.m is a necessary m-file
%-----------------------------------------------------------------

enrgc=zeros(1,-lowest);
enrgd=zeros(1,-lowest);
a=0;
norm=sum(c0 .^2);
for ii=lowest:1:-1
  a=a+1;
  eval(['enrgc(a)=sum(c',num2str(-ii),'.^2);'])
  eval(['enrgd(a)=sum(d',num2str(-ii),'.^2);'])
end
enrgc=enrgc/norm;
enrgc=[enrgc 1];
enrgd=enrgd/norm;
clg
axis([lowest-2 1 0 1.2]);
subplot(211),bar([lowest:0],[enrgc])
title(['Normalized energy of the "c" coefficients  Phase: ',...
      setstr(fliplr(phsvct+48))])
xlabel('Resolution level')
axis([lowest-2 1 0 1.2]);
subplot(212),bar([lowest:-1],[enrgd])
title('Normalized energy of the "d" coefficients')
xlabel('Resolution level')
pause
strplt
axis

%  Energy check of the previous plot

esum=enrgc+[enrgd 0];
clc
dd=['      Verification of the energy in each resolution level     '
    '                                                               '
    '                                                               '
    'Level    Energy in          Energy in C        Difference'
    '          level m-1          level m                       '
    '            c+d                                            '
    '---------------------------------------------------------------'];
disp(dd)
j=-lowest;
for i=1:-lowest
  a=esum(i);b=enrgc(1,i+1);c=1-j;d=abs(a-b);
  fprintf('%2.0f     %17.15f      %17.15f     ',c,a,b)
  fprintf('%17.15f\n',d)
  j=j-1;
end
pause
clc
axis;




%-----------------------------------------------------------------
% recon1d.m  One Dimensional Single Phase Reconsruction
%
% By:    LT  J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:   (408) 646-3044/2772
%
%-----------------------------------------------------------------
% This routine conducts a reconstruction of the the data with the
% proper phase selected between resolution levels.  This is a sub-
% routine for the main wavelet program "wav1d.m".
%
% The following variables are further defined:
%
%     hh & gg       -- reconstruction coefficients
%     lowest        -- defined in "wav1d.m"
%     cwork         -- reconstructed c'values at a particular lvl
%     phsvct        -- phase vector defined in "wav1d.m".
```

```
%
% The following m-files are necessary for proper operation:
%
%     wav1d.m  -- main program
%     rphs1.m  -- reconstruction phase-1 function m-file
%     rphs0.m  -- reconstruction phase-0 function m-file
%     strplt.m  -- storage of the displays
%-----------------------------------------------------------------


clc;
disp(['        '])
q = input('Do you desire to do the Reconstruction Comparison (Y/N)?','s');
clc

if q == 'Y' | q == 'y',
 % generation of the reconstruction coefficients
 hh=fliplr(h);gg=fliplr(g);
 ii=1;
 eval(['cwork=c',num2str(-lowest),';'])
 for lvl=lowest:1:-1
   eval(['dwork=d',num2str(-lvl),';'])
   eval(['cwork=rphs',num2str(phsvct(ii)),'(cwork,dwork,hh,gg);'])
   eval(['a=length(c',num2str(-lvl-1),');'])
   cwork=cwork(1:a);
   ii=ii+1;

   phswrk=phsvct(1:length(phsvct)-1);
   %   Comparison of the Original and the Reconstructed data
   clg
   subplot(211)
    eval(['bar(c',num2str(-lvl-1),')'])
    title(['Original Data     Phase: ',setstr((phswrk+48))])
    xlabel(['Wavelet: ',wwavelet])
   subplot(212)
    bar(cwork);
    title('Reconstructed Data')
    xlabel(['Resolution level ',num2str(lvl+1)])
   pause
   strplt
   clg
   a=eval(['c',num2str(-lvl-1),'-cwork']);
   bar(abs(a));
   title('Absolute Error in the Reconstruction')
   xlabel(['Resolution Level ',num2str(lvl+1)])
   pause
   strplt
   clg
   phswrk=phswrk(1:length(phswrk)-1);
 end
end




function out=rphs0(cwork,dwork,hh,gg)
%*****************************************************************
% rphs0.m    Reconstruction Phase Zero routine
%
% By:    LT J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:  (408) 646-3044/2772
%*****************************************************************
```

```
%
% This routine does a reconstruction of a particular level with a
% phase 0 selection.  The calling program is "wav1d.m".  Input
% arguments are:
%
%       cwork  --  "c" coefficients of a particular level
%       dwork  --  "d" coefficients of a particular level
%       hh     --  reconstruction coeff's for the "c's"
%       gg     --  reconstruciton coeff's for the "d's"
%
% "out" is the output argumemt that equals the c coeff's of the
%  next higher resolution level.
%-----------------------------------------------------------------

L=length(cwork);
x1=[];
x2=[];
for j=1:L
  x1=[x1 cwork(1,j) 0];
  x2=[x2 dwork(1,j) 0];
end
x1=[x1 zeros(1,length(hh)-2)];
x2=[x2 zeros(1,length(gg)-2)];
a=0;
for j=1:length(x1)-(length(hh)-1)
  a=a+1;
  out(1,a)=x1(1,j:j+length(hh)-1)*hh'+x2(1,j:j+length(gg)-1)*gg';
end




function out=rphs1(cwork,dwork,hh,gg)
%*****************************************************************
% rphs1.m    Reconstruction Phase One routine
%
% By:    LT  J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:  (408) 646-3044/2772
%*****************************************************************
%
% This routine does a reconstruction of a particular level with a
% phase 1 selection.  The calling program is "wav1d.m".  Input
% arguments are:
%
%       cwork  --  "c" coefficients of a particular level
%       dwork  --  "d" coefficients of a particular level
%       hh     --  reconstruction coeff's for the "c's"
%       gg     --  reconstruciton coeff's for the "d's"
%
% "out" is the output argumemt that equals the c coeff's of the
%  next higher resolution level.
%-----------------------------------------------------------------

L=length(cwork);
x1=[];
x2=[];
for j=1:L
  x1=[x1 cwork(1,j) 0];
  x2=[x2 dwork(1,j) 0];
end
x1=[0 x1 zeros(1,length(hh)-2)];
x2=[0 x2 zeros(1,length(gg)-2)];
```

```
a=0;
for j=1:length(x1)-(length(hh)-1)
  a=a+1;
  out(1,a)=x1(1,j:j+length(hh)-1)*hh'+x2(1,j:j+length(gg)-1)*gg';
end



%***************************************************************
% plane.m   develop the phase plane diagram
%
% By:    LT  J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:   (408) 646-3044/2772
%***************************************************************
% Phase Plane Diagram for the One Dimensional Single Phase case
% This routine is called by the parent routine "wav1d.m".  All the
% variables defined in "wav1d.m" are used here. New variables:
%
%    c    -- (1-lowest) X LL "c" coefficient matrix
%    d    -- (-lowest) X LL "d" coefficient matrix
%    N,N1 -- variables for the number of contour levels
%
% The main subroutine used is ptzero.m funtion m-file
%-----------------------------------------------------------------

clc
clg

%  Set up the c and d matrices from the "c*" and "d*" variables

c=zeros(-lowest+1,LL);
d=zeros(-lowest,LL);
c(1,1:length(c0))=c0;
clear cwork dwork
for lvl=1:-lowest
  cwork=eval(['ptzero(c',num2str(lvl),',zro(lvl))']);
  dwork=eval(['ptzero(d',num2str(lvl),',zro(lvl))']);
  c(lvl+1,1:length(cwork))=cwork;
  d(lvl,1:length(dwork))=dwork;
end
c=flipud(c.^2);
d=flipud(d.^2);
[i j]=size(c);
x=0:j-1;yc=0:i-1;yd=1:i-1;
mesh(c);
title(['Energy Display of the Approximation   Phase: ',...
      setstr(fliplr(phsvct+48))]);
pause
N=10;
  contour(c,10,x,yc);
  title(['Contour Display of the Approximation   Phase: ',...
        setstr(fliplr(phsvct+48))]);
  ylabel('-n resolution level');
  xlabel('10 contour levels')
pause
clg
q1=input('Do you desire to zoom in on the display (Y/N)?','s');
while q1=='Y' | q1 == 'y',
  q2=input('Do you want to see the plots again (Y/N)?','s');
  if q2=='Y' | q2 == 'y',
   mesh(c);title('Previous Energy Display of the Approximation');
   pause
```

```
    contour(c,N,x,yc);title('PreviousContour Display of the Approximation');
    ylabel('-n resolution level');
    xlabel(['Number of countours: ',num2str(N)])
    pause
   end
   clc
   disp(['   ']')
   disp(['x range: 0 to ',num2str(j-1)])
   disp(['y range: 0 to -',num2str(i-1)]);disp([    ]')
   x1=input('Enter the minimum x-value:')+1;
   x2=input('Enter the maxumum x-value:')+1;
   y1=input('Enter the minimum "magnitude" y-value (least negative):')+1;
   y2=input('Enter the maxumum "magnitude" y-value (most negative):')+1;
   y1=abs(y1);y2=abs(y2);
   y=1:1:i;y=fliplr(y);
   y11=y(y2);
   y22=y(y1);
   clg
   mesh(c(y11:y22,x1:x2))
   title(['Zoomed Energy Display of the Approximation   Phase: ',...
         setstr(fliplr(phsvct+48))]);
   pause
   strplt
   clg

   N1=10;
   while N1<999
     N=N1;
     contour(c(y11:y22,x1:x2),N1,x1:x2,y1-1:y2-1)
     title(['Zoomed Contour Display of the Approximation   Phase: ',...
           setstr(fliplr(phsvct+48))]);
     ylabel('-n resolution level')
     xlabel(['Number of contours: ',num2str(N1)])
     pause
     strplt
     N1=input('Enter the number of contour levels (999 to continue) ');
   end
   q1=input('Zoom in or out Further (Y/N)? ','s');
end
clc

% Now the Detail Signal

mesh(d);
title(['Energy Display of the Detail    Phase: ',setstr(fliplr(phsvct+48))]);
pause
clg

contour(d,10,x,yd);
title(['Contour Display of the Detail   Phase:',setstr(fliplr(phsvct+48))]);
ylabel('-n resolution level');
xlabel('10 contour levels')
pause
clg

clc
q1=input('Do you desire to zoom in on the display (Y/N)?','s');
while q1=='Y' | q1 == 'y',
  q2=input('Do you want to see the plots again (Y/N)?','s');
  if q2=='Y' | q2 =='y',
    mesh(d);title('Previous Energy Display of the Detail');
    pause
    contour(d,N,x,yd);title('PreviousContour Display of the Detail');
    ylabel('-n resolution level');
    xlabel(['Number of contours: ',num2str(N)])
    pause
  end
```

```
    disp(['x range: 0 to ',num2str(j-1)])
    disp(['y range: -1 to -',num2str(i-1)]);disp([     ]')
    x1=input('Enter the minimum x-value:')+1;
    x2=input('Enter the maxumum x-value:')+1;
    y1=input('Enter the minimum "magnitude" y-value (least negative):');
    if y1 == 0, y1=1; end;
    y2=input('Enter the maxumum "magnitude" y-value (most negative):');
    y1=abs(y1);y2=abs(y2);
    y=1:1:i-1;y=fliplr(y);
    y11=y(y2);
    y22=y(y1);
    clg
    mesh(d(y11:y22,x1:x2))
    title(['Zoomed Energy Display of the Detail   Phase: ',...
          setstr(fliplr(phsvct+48))]);
    pause
    strplt
    clg
    while N<999
      contour(d(y11:y22,x1:x2),N,x1:x2,y1:y2)
      title(['Zoomed Contour Display of the Detail  Phase: ',...
            setstr(fliplr(phsvct+48))]);
      ylabel('-n resolution level')
      xlabel(['Number of contours: ',num2str(N)])
      pause
      strplt
      N=input('Enter the number of contour levels (999 to end): ');
    end
    q1=input('Zoom in or out further (Y/N)? ','s');
end




%----------------------------------------------------------------
%  Multiphase routine for The Discrete Wavelet Decomposition Routine
%
% By:    LT J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:   (408) 646-3044/2772
%
%----------------------------------------------------------------
%  This routine does a multiphase discrete wavelet decomposition of
%  a one-dimensional data vector.  This routine requires the following
%  M-files:
%            wav1d.m
%            strplt.m
%            z0pad.m
%-------------------------------------------------------------------
maxSamp=(Nh-1)*(2^(-lowest)-1)+LL;
coeffc=zeros(-lowest+1,maxSamp);
coeffc(-lowest+1,1:LL)=c0;
coeffd=zeros(-lowest,maxSamp);
numcoef=zeros(-lowest+1,1);
%---------------------------------------------------------------
%---------------------------------------------------------------
%  *** The Main Multi-Phase Decomposition Algorithm ***
numcoef(-lowest+1)=LL;
i=1;
for row=-lowest:-1:1
   numcoef(row)=(Nh-1)*(2^i-1)+LL;
   p1=z0pad(coeffc(row+1,1:numcoef(row+1)),h,i);
   coeffc(row,1:length(p1))=p1;
   p2=z0pad(coeffc(row+1,1:numcoef(row+1)),g,i);
   coeffd(row,1:length(p2))=p2;
   clear p1 p2
   i=i+1;
end
%-----------------------------------------------------------------
%     Display of Coefficients at each resolution level
clc
q=input('Do you desire to see values at diff resolution levels (Y/N)? ','s');
if q == 'Y' | q == 'y'
   plot([0:LL-1],c0,'*')
   xlabel(['n sampling points         n*',num2str(1/fsamp),'seconds']);
   ylabel('Amplitude')
   title('Original Data (resolution level 0)  Multiphase case');
   pause
   strplt
   clg
   index=[-lowest:-1:1];
   x1=0;x2=-lowest+1;
   clc
   disp(['Levels to pick are from -1 to ',num2str(lowest)])
   while x1 < 1 | x2 > -lowest
     x1=input('Enter first resolution level of the desired range: ');
     x2=input('Enter second resolution level of the desired range: ');
     x1=abs(x1);x2=abs(x2);
     if x1 > x2, temp=x2; x2=x1; x1=temp; end
   end
   while x1 < =x2
      pts=numcoef(index(x1));
      Tc=coeffc(index(x1),1:pts);
      Td=coeffd(index(x1),1:pts);
      minpts=min(min(Tc),min(Td));ifminpts > 0,minpts=0;end
      maxpts=max(max(Tc),max(Td));ifmaxpts < 0,maxpts=0;end
      if minpts= =0 & maxpts= =0, maxpts=.5;end
      u=[0 pts 1.2*minpts 1.2*maxpts];
      axis(u);
      subplot(211),plot([0:pts-1],coeffc(index(x1),1:pts),'*')
      title(['Multiphase Approximation Coeff at Resolution Level -',num2str(x1)])
      axis(u);
      subplot(212),plot([0:pts-1],coeffd(index(x1),1:pts),'*')
      title('Multiphase Detail Coefficients')
      xlabel(['n sampling points         n*',num2str(1/fsamp),'seconds']);
      pause
      strplt
      clg;clc
      axis;
      x1=x1+1;
   end
end
%-----------------------------------------------------------------
%              Multi-Phase Energy Check
enrg1dmp
%-----------------------------------------------------------------
%                Phase Plane Determination
c=coeffc.^2;
d=coeffd.^2;
[i j]=size(coeffc);
x=0:LL-1;
yc=0:i-1;
yd=1:i-1;
cc=c(1:-lowest+1,1:LL);
clc
mesh(cc);
title('Multiphase Energy Display of the Approximation')
pause
strplt
```

```
clg
N=10;
contour(cc,10,x,yc);
title('Multiphase Contour Display of the Approximation');
ylabel('-n resolution level');
xlabel('10 contour levels')
pause
strplt
clg
q1=input('Do you desire to zoom in/out on the display (Y/N)?','s');
while q1=='Y' | q1 == 'y',
   q2=input('Do you want to see the plots again (Y/N)?','s');
    if q2=='Y' | q2 == 'y',
    mesh(cc)
    title('Previous Multiphase Energy Display of the Approximation');
    pause
    contour(cc,N,x,yc)
    title('Previous Multiphase Contour Display of the Approximation');
    ylabel('-n resolution level');
    xlabel(['Number of contours: ',num2str(N)])
    pause
   end
    clc
    disp(['    '])
    disp(['x range: 0 to ',num2str(j-1)])
    disp(['y range: 0 to -',num2str(i-1)]);disp(['    '])
    x1=input('Enter the minimum x-value:')+1;
    x2=input('Enter the maxumum x-value:')+1;
    y1=input('Enter the minimum "magnitude" y-value (least negative):')+1;
    y2=input('Enter the maxumum "magnitude" y-value (most negative):')+1;
    y1=abs(y1);y2=abs(y2);
    y=1:1:i;y=fliplr(y);
    y11=y(y2);
    y22=y(y1);
    clg
    cc=c(y11:y22,x1:x2);
    mesh(cc)
    title('Zoomed Multiphase Energy Display of the Approximation');
    pause
    strplt
    clg
    N=10;
    N1=10;
    while N1<999
     N=N1;
     contour(cc,N1,x1:x2,y1-1:y2-1)
     title('Zoomed Multiphase Contour Display of the Approximation');
     ylabel('-n resolution level')
     xlabel(['Number of contours: ',num2str(N1)])
     pause
     strplt
     N1=input('Enter the number of contour levels (<RET> to continue) ');
    end
q1=input('Zoom in or out Further (Y/N)? ','s');
end
clc

% Now the Detail Signal
dd=d(1:-lowest,1:LL);
mesh(dd)
title('Multiphase Energy Display of the Detail')
pause
strplt
clg

contour(dd,10,x,yd)
title('Multiphase Contour Display of the Detail');
```

```
ylabel('-n resolution level');
xlabel('10 contour levels')
pause
strplt
clg

clc
q1=input('Do you desire to zoom in/out on the display (Y/N)?','s');
while q1=='Y' | q1 == 'y',
  q2=input('Do you want to see the plots again (Y/N)?','s');
   if q2=='Y' | q2 =='y',
    mesh(dd);title('Previous Multiphase Energy Display of the Detail');
    pause
    contour(dd,N,x,yd);title('PreviousMultiphase Contour Display of the Detail');
    ylabel('-n resolution level');
    xlabel(['Number of contours: ',num2str(N)])
    pause
    end
   disp(['x range: 0 to ',num2str(j-1)])
   disp(['y range: -1 to -',num2str(i-1)]);disp([     ]')
   x1=input('Enter the minimum x-value:')+1;
   x2=input('Enter the maxumum x-value:')+1;
   y1=input('Enter the minimum "magnitude" y-value (least negative):');
   if y1 == 0, y1=1; end;
   y2=input('Enter the maxumum "magnitude" y-value (most negative):');
   y1=abs(y1);y2=abs(y2);
   y=1:1:i-1;y=fliplr(y);
   y11=y(y2);
   y22=y(y1);
   clg
   dd=d(y11:y22,x1:x2);
   mesh(dd)
   title('Zoomed Multiphase Energy Display of the Detail');
   pause
   strplt
   clg
   while N<999
    contour(dd,N,x1:x2,y1:y2)
    title('Zoomed Multiphase Contour Display of the Detail');
    ylabel('-n resolution level')
    xlabel(['Number of contours: ',num2str(N)])
    pause
    strplt
    N=input('Enter the number of contour levels (<RET> to end): ');
   end
q1=input('Zoom in or out further (Y/N)? ','s');
end

%------------------------------------------------
% Multiple Phase Reconstruction

mprecon

%------------------------------------------------



%   The End
```

```
%***************************************************************
% enrg1dmp.m     Determining the Energy in each resolution level
%                      Multi-Phase One Dimensional Case
%
% By:    LT J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:  (408) 646-3044/2772
%***************************************************************
% For the Multi-phase case, the energy is normalized to resolution
% level 0.  All the energies of the "c's" and the "d's" of a
% particular level should add up to the energy of the "c's" of the
% next higher level. The factor of two is accounted for in the Multi-
% Phase case
%
%     enrgc -- energy array of the "c's"
%     enrgd -- energy array of the "d's"
%     norm  -- energy of the data "c0"
%     esum  -- energy sum array (enrgc+enrgd)
%
% The calling routine is "mp.m" .
% wav1d.m and strplt.m are also necessary m-files
%----------------------------------------------------------------

enrgc=zeros(1,-lowest);
enrgd=zeros(1,-lowest);
a=0;
norm=sum(c0.^2);
for row=-lowest:-1:1
  a=a+1;
  enrgc(row)=sum(coeffc(row,:).^2)*2^(-a);
  enrgd(row)=sum(coeffd(row,:).^2)*2^(-a);
end
enrgc=enrgc/norm;
enrgc=[enrgc 1];
enrgd=enrgd/norm;
clg
axis([lowest-2 1 0 1.2]);
subplot(211),bar([lowest:0],[enrgc])
title(['Normalized energy of the "c" coefficients  Multi-Phase Case'])
xlabel('Resolution level')
axis([lowest-2 1 0 1.2]);
subplot(212),bar([lowest:-1],[enrgd])
title('Normalized energy of the "d" coefficients')
xlabel('Resolution level')
pause
strplt
axis

% Energy check of the previous plot

esum=enrgc+[enrgd 0];
clc
dd=['     Verification of the energy in each resolution level     '
    '                                                            '
    '                                                            '
    'Level    Energy in           Energy in C         Difference'
    '         level m-1           level m                       '
    '           c+d                                             '
    '-----------------------------------------------------------'];
disp(dd)
j=-lowest;
for i=1:-lowest
  a=esum(i);b=enrgc(1,i+1);c=1-j;d=abs(a-b);
  fprintf(' %2.0f     %17.15f     %17.15f   ',c,a,b)
  fprintf('%17.15f\n',d)
  j=j-1;
end
pause
clc
clg
axis;



%****************************************************************
% mpplane.m   develop the phase plane diagram
%
% By:    LT  J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:   (408) 646-3044/2772
%****************************************************************
% Phase Plane Diagram for the One Dimensional Single Phase case
% This routine is called by the parent routine "wav1d.m".  All the
% variables defined in "wav1d.m" are used here. New variables:
%
%   c    -- (1-lowest) X LL "c" coefficient matrix
%   d    -- (-lowest) X LL "d" coefficient matrix
%   N,N1 -- variables for the number of contour levels
%
% The main subroutine used is ptzero.m funtion m-file
%-----------------------------------------------------------------

clc
clg

% Set up the c and d matrices from the "c*" and "d*" variables

c=zeros(-lowest+1,LL);
d=zeros(-lowest,LL);
c(1,1:length(c0))=c0;
clear cwork dwork
for lvl=1:-lowest
  cwork=eval(['ptzero(rmpc',num2str(lvl),',2^(lvl)-1)']);
  dwork=eval(['ptzero(rmpd',num2str(lvl),',2^(lvl)-1)']);
  c(lvl+1,1:length(cwork))=cwork;
  d(lvl,1:length(dwork))=dwork;
end
c=flipud(c.^2);
d=flipud(d.^2);



[i j]=size(c);
x=0:j-1;yc=0:i-1;yd=1:i-1;
mesh(c);
title(['*MP* Energy Display of the Approximation   Phase: ',...
      setstr(fliplr(phsvct+48))]);
pause
N=10;
  contour(c,10,x,yc);
  title(['*MP* Contour Display of the Approximation   Phase: ',...
        setstr(fliplr(phsvct+48))]);
  ylabel('-n resolution level');
  xlabel('10 contour levels')
pause
clg
```

```
q1=input('Do you desire to zoom in on the display (Y/N)?','s');
while q1=='Y' | q1 == 'y',
  q2=input('Do you want to see the plots again (Y/N)?','s');
  if q2=='Y' | q2 == 'y',
   mesh(c);
   title('*MP* Previous Energy Display of the Approximation');
   pause
   contour(c,N,x,yc);
   title('*MP* Previous Contour Display of the Approximation');
   ylabel('-n resolution level');
   xlabel(['Number of countours: ',num2str(N)])
   pause
  end
  clc
  disp(['   '])
  disp(['x range: 0 to ',num2str(j-1)])
  disp(['y range: 0 to -',num2str(i-1)]);disp([    ])
  x1=input('Enter the minimum x-value:')+1;
  x2=input('Enter the maxumum x-value:')+1;
  y1=input('Enter the minimum "magnitude" y-value (least negative):')+1;
  y2=input('Enter the maxumum "magnitude" y-value (most negative):')+1;
  y1=abs(y1);y2=abs(y2);
  y=1:1:i;y=fliplr(y);
  y11=y(y2);
  y22=y(y1);
  clg
  mesh(c(y11:y22,x1:x2))
  title(['*MP* Zoomed Energy Display of the Approximation   Phase: ',...
       setstr(fliplr(phsvct+48))]);
  pause
  strplt
  clg

  N1=10;
  while N1<999
    N=N1;
    contour(c(y11:y22,x1:x2),N1,x1:x2,y1-1:y2-1)
    title(['*MP* Zoomed Contour Display of the Approximation   Phase: ',...
         setstr(fliplr(phsvct+48))]);
    ylabel('-n resolution level')
    xlabel(['Number of contours: ',num2str(N1)])
    pause
    strplt
    N1=input('Enter the number of contour levels (999 to continue) ');
  end
  q1=input('Zoom in or out Further (Y/N)? ','s');
end
clc

% Now the Detail Signal

mesh(d);
title(['*MP* Energy Display of the Detail    Phase: ',...
  setstr(fliplr(phsvct+48))]);
pause
clg

contour(d,10,x,yd);
title(['*MP* Contour Display of the Detail   Phase:',...
  setstr(fliplr(phsvct+48))]);
ylabel('-n resolution level');
xlabel('10 contour levels')
pause
clg

clc
q1=input('Do you desire to zoom in on the display (Y/N)?','s');
```

```
while q1=='Y' | q1 == 'y',
  q2=input('Do you want to see the plots again (Y/N)?','s');
  if q2=='Y' | q2 =='y',
    mesh(d);title('Previous Energy Display of the Detail');
    pause
    contour(d,N,x,yd);title('PreviousContour Display of the Detail');
    ylabel('-n resolution level');
    xlabel(['Number of contours: ',num2str(N)])
    pause
  end
  disp(['x range: 0 to ',num2str(j-1)])
  disp(['y range: -1 to -',num2str(i-1)]);disp([     ])
  x1=input('Enter the minimum x-value:')+1;
  x2=input('Enter the maxumum x-value:')+1;
  y1=input('Enter the minimum "magnitude" y-value (least negative):');
  if y1 == 0, y1=1; end;
  y2=input('Enter the maxumum "magnitude" y-value (most negative):');
  y1=abs(y1);y2=abs(y2);
  y=1:1:i-1;y=fliplr(y);
  y11=y(y2);
  y22=y(y1);
  clg
  mesh(d(y11:y22,x1:x2))
  title(['*MP* Zoomed Energy Display of the Detail   Phase: ',...
        setstr(fliplr(phsvct+48))]);
  pause
  strplt
  clg
  while N<999
    contour(d(y11:y22,x1:x2),N,x1:x2,y1:y2)
    title(['*MP* Zoomed Contour Display of the Detail  Phase: ',...
          setstr(fliplr(phsvct+48))]);
    ylabel('-n resolution level')
    xlabel(['Number of contours: ',num2str(N)])
    pause
    strplt
    N=input('Enter the number of contour levels (999 to end): ');
  end
  q1=input('Zoom in or out further (Y/N)? ','s');
end
```

```
function x=z0pad(Data,h,reslvl)
%*****************************************************************
% z0pad.m   *** For the Multi-phase decomposition routine ***
%                    One Dimensional Case
%
% By:    LT J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:   (408) 646-3044/2772
%*****************************************************************
%
% This routine makes a row vector of the lower coefficients of
% a particular resolution level.  Use for only the Multi-phase
% decomposition One-Dimensional case !!
%
%         Data is the data vector
%         h is either the h coeff or the g coeff vector
%         reslvl is the resolution level
%-----------------------------------------------------------------
```

```
m=realvl;
L=length(h);
j=L;
x=0;
for i=1:L
  x=x+h(j)*[zeros(1,(i-1)*2^(m-1))Data zeros(1,(j-1)*2^(m-1))];
  j=j-1;
end
```

```
function binary=num2bin(number,length1)
%----------------------------------------------------------------
% NUM2BIN.M        Number to Binary conversion
%
% By:    LT J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:   (408) 646-3044/2772
%
%----------------------------------------------------------------
% This routine conducts a conversion of a base 10 integer to a base 2
% data array i.e.  [1 0 1 0 1 1]
%
% The following variables are further defined:
%
%      binary    -- binary output data array
%      length1   -- input desired length of the display
%      number    -- base 10 integer
%
% The following m-files are necessary for proper operation:
%
%      wav1d.m  -- main program (great grandparent routine)
%      mp.m      -- grandparent routine
%      mprecon.m -- multiphase reconstuction calling routine
%----------------------------------------------------------------

binary=[];
while number>.5
   number=number/2;
   if number-fix(number)==.5
      number=number-.5;
      binary=[1 binary];
    else
      binary=[0 binary];
    end
end

if length(binary)<length1
    binary=[zeros(1,abs(length1-length(binary)))binary];
end
```

```
function num=bin2num(bin)
%---------------------------------------------------------------
% BIN2NUM.M       Binary to number conversion
%
% By:    LT J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:   (408) 646-3044/2772
%
%------------------------------------------------------------------
% This routine conducts a conversion of a base 2 data array
% i.e.  [1 0 1 0 1 1] to a base 10 integer
%
% The following variables are further defined:
%
%     bin    -- binary output data array
%     num    -- base 10 integer
%
% The following m-files are necessary for proper operation:
%
%     wav1d.m   -- main program (great grandparent routine)
%     mp.m      -- grandparent routine
%     mprecon.m -- multiphase reconstuction calling routine
%------------------------------------------------------------------


num=0;
a=length(bin);
power=a;
for ii=1:a
   num=num+bin(ii)*2^(power-ii);
end
```

B. TWO-DIMENSIONAL ROUTINES

```
%*****************************************************************
% WAV2D.M                                    15 SEP 92
%     Wavelet decomposition algorithm for the two-dimensional case
%     for either the single selectable (of 4 possible) or the
%     multiple phase case.
%
% By:   LT J. E. Legaspi
%       Professor Lam
%       US Naval Postgraduate School, Monterey CA
%       e-mail: legaspi@ece.nps.navy.mil
%               lam@ece.nps.navy.mil
%       Phone:  (408) 646-3044/2772
%*****************************************************************
% This routine does a two-dimensional dicrete wavelet decomposition
% of a two-dimensional data array. The following routines are also
% necessary for the proper operation of the algorithm:
%
%       chkputer.m -- checks for the minimum hardware
%       mp2d.m     -- 2-D multiple phase decomposition
%       ptzro2d.m  -- 2-D row & col zero padding
%       strplt.m   -- metafiles of selected plots
%       hcoefin    -- routine for selected h coefficients
%       phas ?? .m -- routines for phase 00,01,10,11 decomp
%       enrg2d.m   -- energy check for the 2-D case
%       recon2d.m  -- 2-D reconstruction
%-----------------------------------------------------------------
% Variables:
%
%    pltcnt      - plot counter for hard copy displays
%    Imagestr    - string variable of the input data
%    c0          - resolution level 0 data input
%    c1 to cN    - corresponds to the c coeff level -1 to -N
%    d11 to d1N  - corresponds to the d1 coeff's
%    d21 to d2N  - corresponds to the d2 coeff's
%    d31 to d3N  - corresponds to the d3 coeff's
%    Hh,Hv,Gh,Gv - h and g coeff's in the horiz & vert directions
%    HhHv,HhGv,
%    GhHv,GhGv   - 2-D decomposition masks
%    lowest      - lowest resolution level (a neg number)
%    phsvct?     - phase vector for either the h or v direction
%
%-----------------------------------------------------------------

%----------------------------------------
% initialization of the counters and plots

clg
axis('normal');
hold off;
pltcnt=0;
chkputer
clc
%----------------------------------------
%----------------------------------------
% 2-D data input routine

Imagestr=input('Enter the name of the image data: ','s');
c0=eval(Imagestr);
[i1 j1]=size(c0);
  if      j1<=1 | i1<=1
          disp('Bad Input Data. Restart the Program')
          return
  end
```

```
clc
%----------------------------------------
%----------------------------------------
% 2-D wavelet coefficient input routine
hcoefin
clg
%----------------------------------------
%----------------------------------------
% Branch for the 2-D multiphase case

disp(['      '])
q=input('Do you desire the 2-D multiphase decomposition only? ','s');
if q=='y' | q=='Y'
  mp2d
  return
end
%----------------------------------------
%-----------------------------------------------------------------
% Initialization for the decomposition routine
%    of the single selectable phase
%
% initializing Cout for the decomposition routine. Cout along
% with D1out, D2out and D3out are intermediate working
% variables for the determining the coefficients for a
% particular resolution level and phase. 'phas??.m' are
% one of for possible phase decomposition routines for the
% two dimensional case.

Cout=c0;
subplot(211),mesh(c0),title('3-DPlot of the Data Array')
subplot(212),contour(c0),title('ContourDisplay of the Data Array')
pause
strplt
clg
clc

disp(['      '])
q=['Enter the number of resolution levels desired'
   'for decomposition of the image:              '];
disp(q)
lowest=input(': >  ');
clc
lowest=-abs(lowest);

phsvcth=[];
phsvctv=[];

% The actual decomposition for the single
%      selectable phase case

for lvl=-1:-1:lowest

   clc
   q=['Enter the desired phase decomposition:  '
     '                                       '
     'A)  Horizontal Phase 0, Vertical Phase 0'
     'B)  Horizontal Phase 0, Vertical Phase 1'
     'C)  Horizontal Phase 1, Vertical Phase 0'
     'D)  Horizontal Phase 1, Vertical Phase 1'
     '                                       '];
  disp(['      '])
  disp(q)
  phase=input(' : ','s');

 if    phase=='A' | phase=='a'
       phsvcth=[0 phsvcth];
       phsvctv=[0 phsvctv];
```

```
        phas00
  elseif phase=='B' | phase=='b'
        phsvcth=[0 phsvcth];
        phsvctv=[1 phsvctv];
        phas01
  elseif phase=='C' | phase=='c'
        phsvcth=[1 phsvcth];
        phsvctv=[0 phsvctv];
        phas10
  elseif phase=='D' | phase=='d'
        phsvcth=[1 phsvcth];
        phsvctv=[1 phsvctv];
        phas11
  else
        disp('Error')
        return
  end

end

% clean up for better memory utilization
%       This is primarily for 386 MATLAB
clear wrkspc D1out D2out D3out Cout wrk a b q qq qqq q1
pack
%--------------------------------------------------------------------
%--------------------------------------------
% Energy check for the 2-D single selectable phase case
enrg2d
% clean up for better memory utilization
%       This is primarily for 386 MATLAB
clear E k lvl a b c j
pack
%--------------------------------------------
%--------------------------------------------
% Display of the phase sequence in reverse bit order as per the notes
clc
disp(['     '])
disp('Order of the Selected Phases from the highest to the lowest level')
disp(' ')
disp('Horizontal: Reading left to right corresponds to going from the')
disp('   higher resolution to the lower resolution level   ')
disp([' '])
disp(fliplr(phsvcth))
disp(['     '])
disp('Vertical: Reading left to right corresponds to going from the')
disp('   higherer resolution to lower resolution level   ')
disp([' '])
disp(fliplr(phsvctv))
disp(['     '])
disp('<RET> to continue')
pause
clc
%--------------------------------------------
%--------------------------------------------
% Conduct the recomposition routine
recon2d
%--------------------------------------------
clc
disp(['    '])
disp('        ROUTINE  COMPLETE')

%   The End of WAV2D.M
```

```
%******************************************************************
% HCOEFIN.m    2-D wavelet coefficient input
%
% By:   LT J. E. Legaspi
%       Professor Lam
%       US Naval Postgraduate School, Monterey CA
%       e-mail: legaspi@ece.nps.navy.mil
%               lam@ece.nps.navy.mil
%       Phone:  (408) 646-3044/2772
%******************************************************************
% This routine obtains the desired wavelet h coefficients for both
% the horizontal and vertical directions. The wavelets are assumed
% separable.
%------------------------------------------------------------------
% Variables:
%
%   Hh,Hv,Gh,Gv -   h and g coeff's in the horiz & vert directions
%   HhHv,HhGv,
%   GhHv,GhGv   -   2-D decomposition masks
%   wavelet?    -   string variable in the h or v directions
%------------------------------------------------------------------
%-----------------------------------------------
% Obtaining the horizontal h coefficients
disp(['        '])
q1=['Enter the number for the desired choice     '
    'for the wavelet in the HORIZONTAL direction:'
    '                                            '
    '(1) Haar h coefficients                     '
    '(2) Daubechies h coefficients               '
    '(3) Own set of h coefficients               '
    '                                            '];
disp(q1)
qqq=input(' ');
 if     qqq==3,
        Hh=input('Enter your own set of H coefficients');
        a=sum(Hh);b=Hh*Hh';
        while a~=1 & b~=.5
         disp('Your h coefficients are NOT compactly supported!')
         Hh=input('Re-enter the h coefficients or ctrl-C to stop');
        end
        waveleth=['Own'];
 elseif qqq==2,
        qq=input('How many taps (2-10)? ');
        Hh=daubdata(qq);
        waveleth=['Daubechies ',num2str(qq),'-tap'];
 else    Hh=[.5 .5];
        waveleth=['Haar'];
        qqq=1;
 end
clc
%-----------------------------------------------
% Now for the vertical case
disp(['      '])
q=input('Do you desire the same wavelet in the vertical direction?','s');
if q=='Y' | q == 'y'
  Hv=Hh;
  waveletv=waveleth;
else
 disp(['        '])
 q1=['Enter the number for the desired choice    '
     'for the wavelet in the VERTICAL direction: '
     '                                           '
     '(1) Haar h coefficients                    '
     '(2) Daubechies h coefficients              '
     '(3) Own set of h coefficients              '
     '                                           '];
 disp(q1)
```

```
qqq=input(' ');
  if      qqq==3,
          Hv=input('Enter your own set of H coefficients');
          a=sum(Hh);b=Hh*Hh';
          while a~=1 & b~=.5
           disp('Your h coefficients are NOT compactly supported!')
           Hh=input('Re-enter the h coefficients or ctrl-C to stop');
          end
          waveletv=['Own'];
  elseif  qqq==2,
          qq=input('How many taps (2-10)? ');
          Hv=daubdata(qq);
          waveletv=['Daubechies',num2str(qq),'-tap'];
  else    Hv=[.5 .5];
          waveletv=['Haar'];
          qqq=1;
  end
 clc
end
%-----------------------------------------------
%-----------------------------------------------
%     Generating the g coefficients
Gh=fliplr(Hh);
for i=2:2:length(Hh)
  Gh(1,i)=-Gh(1,i);
end
Gv=fliplr(Hv);
for i=2:2:length(Hv)
   Gv(1,i)=-Gv(1,i);
end
%-----------------------------------------------
%-----------------------------------------------
%     Generating the 2-D masks for the decomposition
Hv=Hv';
Gv=Gv';
%the 2* factor here reduces the number of multiplication2 for
%each lower level coefficients
HhHv=2*Hv*Hh;
HhGv=2*Gv*Hh;
GhHv=2*Hv*Gh;
GhGv=2*Gv*Gh;
NHh=length(Hh);
NHv=length(Hv);
%   End




%*********************************************************************
% PHAS00.M    2-D phase 00 decomposition of the data array
%
%
% By:    LT J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:  (408) 646-3044/2772
%*********************************************************************
% This routine does a phase 00 decomposition given the input array - Cout.
%    horiz: 0      vert: 0
%
% Variables:
%          Cout, D1out-D3out  -  working variables for the coefficients
%          wrkspc             -  working array for the c coeff's
```

```
%-----------------------------------------------------------------------
% Required m-files:
%                  wav2d.m    -  calling program
%                  decomplt.m -  plotting routine for the res level
%
%-----------------------------------------------------------------------
clc
[rows cols]=size(Cout);
 %check if the row is odd, if it is, pad with one zero
 if rem(rows/2,floor(rows/2)) == 0
   Cout=[Cout;zeros(1,cols)];
   rows=rows+1;
 end
 %do the same with the columns
 if rem(cols/2,floor(cols/2)) == 0
   Cout=[Cout zeros(rows,1)];
   cols=cols+1;
 end
 % generate the workspace
 wrkspc=[zeros(NHv-1,cols+2*NHh-3);zeros(rows,NHh-1)Cout zeros(rows,NHh-2);...
        zeros(NHv-1,cols+2*NHh-3)];
 clear Cout D1out D2out D3out
   % Now operate on the wrkspc with the decomposition masks
  a=0;
  for row1=1:2:rows+NHv-2
    a=a+1;
    b=0;
    for col1=1:2:cols+NHh-2
     b=b+1;
     wrk=wrkspc(row1:row1+NHv-1,col1:col1+NHh-1);
     Cout(a,b)= sum(sum(HhHv .* wrk));
     D1out(a,b)=sum(sum(HhGv.* wrk));
     D2out(a,b)=sum(sum(GhHv.* wrk));
     D3out(a,b)=sum(sum(GhGv.* wrk));
    end
  end
% Store the working variables with the appropriate name
eval(['c',num2str(abs(lvl)),'=Cout;'])
eval(['d1',num2str(abs(lvl)),'=D1out;'])
eval(['d2',num2str(abs(lvl)),'=D2out;'])
eval(['d3',num2str(abs(lvl)),'=D3out;'])
clg
% Display the variables C, D1, D2 and D3 for the resolution level
decomplt




%***********************************************************************
% PHAS10.M     2-D phase 01 decomposition of the data
%
%
% By:    LT J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:   (408) 646-3044/2772
%***********************************************************************
% This routine does a phase 10 decomposition given the input array - Cout.
%    horiz: 1      vert: 0
%
% Variables:
%          Cout, D1out-D3out  -  working variables for the coefficients
%          wrkspc             -  working array for the c coeffs
```

```
%-----------------------------------------------------------------------
% Required m-files:
%                wav2d.m    - calling program
%                decomplt.m - plotting routine for the res level
%-----------------------------------------------------------------------
[rows cols]=size(Cout);
%check if the row is even, if it is, pad with one zero
if rem(rows/2,floor(rows/2)) == 0
   Cout=[Cout;zeros(1,cols)];
   rows=rows+1;
end
%odd column check
if rem(cols/2,floor(cols/2)) ~= 0
   Cout=[Cout zeros(rows,1)];
   cols=cols+1;
end
% generate the workspace
wrkspc=[zeros(NHv-1,cols+2*NHh-4);zeros(rows,NHh-2)Cout zeros(rows,NHh-2);...
        zeros(NHv-2,cols+2*NHh-4)];
clear Cout D1out D2out D3out
a=0;
for row1=1:2:rows+NHv-2
  a=a+1;
  b=0;
  for col1=1:2:cols+NHh-2
   b=b+1;
   wrk=wrkspc(row1:row1+NHv-1,col1:col1+NHh-1);
   Cout(a,b)= sum(sum(HhHv .* wrk));
   D1out(a,b)=sum(sum(HhGv.* wrk));
   D2out(a,b)=sum(sum(GhHv.* wrk));
   D3out(a,b)=sum(sum(GhGv.* wrk));
  end
 end
lvl2=abs(lvl);
eval(['c',num2str(lvl2),'=Cout;'])
eval(['d1',num2str(lvl2),'=D1out;'])
eval(['d2',num2str(lvl2),'=D2out;'])
eval(['d3',num2str(lvl2),'=D3out;'])
clg
decomplt



%***********************************************************************
% PHAS01.M     2-D phase 01 decomposition of the data array
%
%
% By:    LT J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail: legaspi@ece.nps.navy.mil
%                lam@ece.nps.navy.mil
%        Phone:  (408) 646-3044/2772
%***********************************************************************
% This routine does a phase 01 decomposition given the input array - Cout.
%    horiz: 0      vert: 1
%
% Variables:
%            Cout, D1out-D3out  -  working variables for the coefficients
%            wrkspc             -  working array for the c coeff's

%-----------------------------------------------------------------------
% Required m-files:
%                wav2d m    - calling program
%                decomplt.m - plotting routine for the res level
```

```
%------------------------------------------------------------------------
clc
[rows cols]=size(Cout);
%check if the row is odd, if it is, pad with one zero
if rem(rows/2,floor(rows/2)) ~ = 0
   Cout=[Cout;zeros(1,cols)];
   rows=rows+1;
end
%check if the columns are even
if rem(cols/2,floor(cols/2)) = = 0
   Cout=[Cout zeros(rows,1)];
   cols=cols+1;
end
% generate the workspace
wrkspc=[zeros(NHv-2,cols+2*NHh-3);zeros(rows,NHh-1)Cout zeros(rows,NHh-2);...
       zeros(NHv-2,cols+2*NHh-3)];
clear Cout D1out D2out D3out
 a=0;
 for row1=1:2:rows+NHv-2
   a=a+1;
   b=0;
   for col1=1:2:cols+NHh-2
    b=b+1;
    wrk=wrkspc(row1:row1+NHv-1,col1:col1+NHh-1);
    Cout(a,b)= sum(sum(HhHv.* wrk));
    D1out(a,b)=sum(sum(HhGv.* wrk));
    D2out(a,b)=sum(sum(GhHv.* wrk));
    D3out(a,b)=sum(sum(GhGv.* wrk));
   end
  end
lvl2=abs(lvl);
eval(['c',num2str(lvl2),'=Cout;'])
eval(['d1',num2str(lvl2),'=D1out;'])
eval(['d2',num2str(lvl2),'=D2out;'])
eval(['d3',num2str(lvl2),'=D3out;'])
clg
decomplt




%*********************************************************************
% PHAS11.M     2-D phase 11 decomposition of the data array
%
%
% By:    LT  J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:  (408) 646-3044/2772
%*********************************************************************
% This routine does a phase 11 decomposition given the input array - Cout.
%   horiz: 1      vert: 1
%
% Variables:
%           Cout, D1out-D3out  -  working variables for the coefficients
%           wrkspc             -  working array for the c coeff's

%------------------------------------------------------------------------
% Required m-files:
%                 wav2d.m    -  calling program
%                 decomplt.m -  plotting routine for the res level
%
%------------------------------------------------------------------------
clc
```

```
[rows cols]=size(Cout);

%check of the row is odd, if it is, pad with one zero
if rem(rows/2,floor(rows/2)) ~ = 0
   Cout=[Cout;zeros(1,cols)];
   rows=rows+1;
end
 %do the same with the columns
if rem(cols/2,floor(cols/2)) ~ = 0
   Cout=[Cout zeros(rows,1)];
   cols=cols+1;
end
 % generate the workspace
wrkspc=[zeros(NHv-2,cols+2*NHh-4);zeros(rows,NHh-2)Cout zeros(rows,NHh-2);...
        zeros(NHv-2,cols+2*NHh-4)];
clear Cout D1out D2out D3out
a=0;
for row1=1:2:rows+NHv-2
  a=a+1;
  b=0;
  for col1=1:2:cols+NHh-2
   b=b+1;
   wrk=wrkspc(row1:row1+NHv-1,col1:col1+NHh-1);
   Cout(a,b)= sum(sum(HhHv.* wrk));
   D1out(a,b)=sum(sum(HhGv.* wrk));
   D2out(a,b)=sum(sum(GhHv.* wrk));
   D3out(a,b)=sum(sum(GhGv.* wrk));
  end
 end
lvl2=abs(lvl);
eval(['c',num2str(lvl2),'=Cout;'])
eval(['d1',num2str(lvl2),'=D1out;'])
eval(['d2',num2str(lvl2),'=D2out;'])
eval(['d3',num2str(lvl2),'=D3out;'])
clg
decomplt




%*******************************************************************
% DECOMPLT.M  displays the coefficients for a particular level
%
% By:    LT  J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:   (408) 646-3044/2772
%*******************************************************************
% Variables: as defined in WAV2D.M and PHAS??.M
%
% Required m-files:
%                  WAV2D.M     parent routine
%                  PHAS??.M    one of four possible calling routines
%                  STRPLT.M    metafile storage of selected plots
%-------------------------------------------------------------------
 clg
 subplot(221),contour(Cout);xlabel('Cm');
 title(['wavelet horiz: ',waveleth]);
 subplot(222),contour(D1out);xlabel('D1m')
 title(['wavelet vert: ',waveletv])
 subplot(223),contour(D2out);title('D2m')
 xlabel(['Horiz phase: ',setstr(fliplr(phsvcth+48)),' Vert phase: ',...
       setstr(fliplr(phsvctv+48))])
 subplot(224),contour(D3out);title('D3m')
```

```
xlabel(['Resolution Level ',num2str(lvl)])
pause
strplt
clg
subplot(221),mesh(Cout);xlabel('Cm')
title(['wavelet horiz: ',waveleth]);
subplot(222),mesh(D1out);xlabel('D1m')
title(['wavelet vert: ',waveletv])
subplot(223),mesh(D2out);title('D2m')
xlabel(['Horiz phase: ',setstr(fliplr(phsvcth+48)),' Vert phase: ',...
      setstr(fliplr(phsvctv+48))])
subplot(224),mesh(D3out);title('D3m')
xlabel(['Resolution Level ',num2str(lvl)])
pause
strplt

%   END




%****************************************************************
% enrg2d.m     Determining the Energy in each resolution level
%                        Two Dimensional Case
%
% By:    LT  J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:   (408) 646-3044/2772
%****************************************************************
% For the single phase case, the energy is normalized to resolution
% level 0.  All the energies of the "c's" and the "d's" of a
% particular level should add up to the energy of the "c's" of the
% next higher level. The calling routine is "wav2d.m" and "strplt.m"
% is a necessary m-file.
%----------------------------------------------------------------
%   Variables (other than defined in mp2d.m):
%
%     E    - energy array for [c,d1,d2,d3] for each level
%     esum -  total energy of each level: [c(m)+d1+d2+d3]=c(m+1)
%----------------------------------------------------------------
% Initialization of the energy sum for resolution level 0
E=zeros(-lowest+1,4);
E(1,1)=sum(sum(c0.^2));
lvl=-1;
k=1;
% Calculate the energies for each resolution level
while lvl>=lowest
  E(k+1,1)=sum(sum((eval(['c',num2str(abs(lvl))])).^2));
  E(k+1,2)=sum(sum((eval(['d1',num2str(abs(lvl))])).^2));
  E(k+1,3)=sum(sum((eval(['d2',num2str(abs(lvl))])).^2));
  E(k+1,4)=sum(sum((eval(['d3',num2str(abs(lvl))])).^2));
  lvl=lvl-1;
  k=k+1;
end
E=E/E(1,1);  % normalize the energy to level 0
E=flipud(E);  % top row is the lowest resolution
% Display of the energies by coefficients
clg
k=[lowest-1 1 0 1.1];
axis(k);subplot(221),bar(lowest:1:0,E(:,1))
        xlabel('c')
        title(['wavelet horiz: ',waveleth])
```

```
axis(k);subplot(222),bar(lowest:1:0,E(:,2))
       xlabel('d1')
       title(['wavelet vert: ',waveletv])
axis(k);subplot(223),bar(lowest:1:0,E(:,3))
       title('d2')
       xlabel('single phase case')
axis(k);subplot(224),bar(lowest:1:0,E(:,4))
       title('d3')
pause
strplt
clg
clc
E=E';
esum=sum(E);
disp(['     '])
dd=['     Verification of the energy in each resolution level     '
    '                                                          '
    '                                                          '
    'Level    Energy in            Energy in C          Difference'
    '  m      level m-1            level m                    '
    '         c+d1+d2+d3                                       '
    '-------------------------------------------------------------'];
disp(dd)
j=-lowest;
for i=1:-lowest
  a=esum(i);b=E(1,i+1);c=-j+1;d=abs(a-b);
  fprintf('%2.0f     %17.15f     %17.15f     ',c,a,b)
  fprintf('%17.15f\n',d)
  j=j-1;
end
pause
clc
%            End




%*****************************************************************
% recon2d.m     2-D reconstruction algorithm for the single
%               selectable phase case
%
%
% By:    LT  J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:   (408) 646-3044/2772
%-----------------------------------------------------------------
%  Variables (other than defined in wav2d.m):
%
%     HvHh,HvGh  -  reconstrution masks
%     GvHh,GvGh
%     which1     -  phase decomposition for the particular level
%     cwork      -  reconstructed c coefficient array for a level
%
%   Required routines:
%                    wav2d.m
%                    strplt.m
%                    mmlt??.m  -- ?? = 00,01,10,11 for the
%                                 mask multiplication
%
%-----------------------------------------------------------------
clc
disp(['     '])
q=input('Do you desire to do the reconstruction (Y/N)? ','s');
```

```
if q=='y' |q=='Y'
% The following four lines generate the reconstruction masks
HvHh=fliplr(flipud(HhHv));
HvGh=fliplr(flipud(GhHv));
GvHh=fliplr(flipud(HhGv));
GvGh=fliplr(flipud(GhGv));
disp(['    '])
q=input('Do you desire to see the masks? ','s');
if q=='Y' | q=='y'
  % We will border the arrays with zeros for a better mesh plot
  [aa bb]=size(HhHv);
  disp(' ')
   disp('HhHv  decomposition')
   disp(HhHv)
   subplot(121)
     axis('square')
     mesh([zeros(1,bb+2);zeros(aa,1)HhHv zeros(aa,1);zeros(1,bb+2)])
     title('HhHv decomposition mask')
   disp(' ')
   disp('HvHh  reconstruction')
   disp(HvHh)
   subplot(122)
     axis('square')
     mesh([zeros(1,bb+2);zeros(aa,1)HvHh zeros(aa,1);zeros(1,bb+2)])
     title('HvHh reconstruction mask')
   pause
   strplt
   clg
  disp(' ')
   disp('HhGv  decomposition')
   disp(HhGv)
   subplot(121)
     axis('square')
     mesh([zeros(1,bb+2);zeros(aa,1)HhGv zeros(aa,1);zeros(1,bb+2)])
     title('HhGv decomposition mask')
   disp(' ')
   disp('GvHh  reconstruction')
   disp(GvHh)
   subplot(122)
     axis('square')
     mesh([zeros(1,bb+2);zeros(aa,1)GvHh zeros(aa,1);zeros(1,bb+2)])
     title('GvHh reconstruction mask')
   pause
   strplt
   clg
  disp(' ')
   disp('GhHv  decomposition')
   disp(GhHv)
   subplot(121)
     axis('square')
     mesh([zeros(1,bb+2);zeros(aa,1)GhHv zeros(aa,1);zeros(1,bb+2)])
     title('GhHv decomposition mask')
   disp(' ')
   disp('HvGh  reconstruction')
   disp(HvGh)
   subplot(122)
     axis('square')
     mesh([zeros(1,bb+2);zeros(aa,1)HvGh zeros(aa,1);zeros(1,bb+2)])
     title('HvGh reconstruction mask')
    pause
    strplt
    clg
  disp(' ')
   disp('GhGv  decomposition')
   disp(GhGv)
   subplot(121)
     axis('square')
```

```
      mesh([zeros(1,bb+2);zeros(aa,1)GhGv zeros(aa,1);zeros(1,bb+2)])
      title('GhGv decomposition mask')
    disp(' ')
    disp('GvGh  reconstruction')
    disp(GvGh)
    subplot(122)
      axis('square')
      mesh([zeros(1,bb+2);zeros(aa,1)GvGh zeros(aa,1);zeros(1,bb+2)])
      title('GvGh reconstruction mask')
    pause
    strplt
    clg
end
clear aa bb
axis('normal')
clc
disp(['       '])
disp('    NOW DOING THE RECONSTRUCTION')
%---------------------------------------------
% start at the lowest level
%  and calculate the c's for the next level
lowest=-abs(lowest);
lvl=lowest;
cwork=eval(['c',num2str(abs(lowest))]);
aa=0;
while lvl<0
 aa=aa+1;
 [a b]=size(eval(['c',num2str(abs(lvl+1))]));
 d1=eval(['d1',num2str(abs(lvl))]);
 d2=eval(['d2',num2str(abs(lvl))]);
 d3=eval(['d3',num2str(abs(lvl))]);
 which1=[num2str(phsvcth(aa)),num2str(phsvctv(aa))];
 eval(['tc=mmlt',which1,'(cwork,HvHh);'])
 eval(['td1=mmlt',which1,'(d1,GvHh);'])
 eval(['td2=mmlt',which1,'(d2,HvGh);'])
 eval(['td3=mmlt',which1,'(d3,GvGh);'])
 clear cwork d1 d2 d3
 cwork=tc(1:a,1:b)+td1(1:a,1:b)+td2(1:a,1:b)+td3(1:a,1:b);
 clear tc td1 td2 td3 a b which1
 clg
 axis('square')
 % plot the result of the c's for comparison
 subplot(121),contour(cwork);
   title('Reconstructed C array')
  subplot(122),contour(eval(['c',num2str(abs(lvl)-1)]));
   title(['Actual C array'])
   xlabel(['Resolution Level ',num2str(lvl+1)])
 pause
 strplt
 clg
 subplot(121),mesh(cwork);
   title('Reconstructed C array')
 subplot(122),mesh(eval(['c',num2str(abs(lvl)-1)]));
   title(['Actual C array'])
   xlabel(['Resolution Level ',num2str(lvl+1)])
 pause
 strplt
 clg
 errr=cwork-eval(['c',num2str(abs(lvl+1))]);
 if max(max(abs(errr)))~=0
   mesh(abs(errr))
   title(['Absolute error from ',num2str(lvl),' to ',num2str(lvl+1)])
   xlabel(['Maximum Error Value: ',num2str(max(max(abs(errr))))])
   pause
   strplt
 else
   clc
   clg
   disp(['      '])
   disp('         *** NO ERROR IN THE RECONSTRUCTION ***')
   disp(['  '])
   disp('           NOW CALCULATING THE NEXT RECONSTRUCTION')
   disp(['                  FROM LEVEL 'num2str(lvl+1),' TO ',...
      num2str(lvl+2)])
  end
lvl=lvl+1;
end
axis('normal')
clc
end
```

```
function x=mmlt00(array,mask)
%*********************************************************************
% MMLT00.M    mask multiplication and summation for the 2-D phase 00
%             reconstruction
%
% By:    LT  J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:   (408) 646-3044/2772
%*********************************************************************
% !!!!!!!! This routine is only for PHASE-00 !!!!!!!!!
% Required M=files:
%                wav2d.m  -- grandparent routine
%                recon2d.m -- parent routine
%                ptzro2d.m -- puts n zeros rows & cols btwn coeff's
%--------------------------------------------------------------------
array=ptzro2d(array,1);
 % generate the workspace
[NHv NHh]=size(mask);
[row col]=size(array);
array=[array zeros(row,NHh-2)];cols=col+NHh-2;
array=[array; zeros(NHv-2,cols)];
a=0;
 for row1=1:row-1
   a=a+1;
   b=0;
   for col1=1:col-1
    b=b+1;
    wrk=array(row1:row1+NHv-1,col1:col1+NHh-1);
    x(a,b)=sum(sum(mask.*wrk));
   end
 end
```

```
function x=mmlt10(array,mask)
%*******************************************************************
% MMLT10.M    mask multiplication and summation for the 2-D phase 10
%             reconstruction
%
% By:   LT J. E. Legaspi
%       Professor Lam
%       US Naval Postgraduate School, Monterey CA
%       e-mail: legaspi@ece.nps.navy.mil
%               lam@ece.nps.navy.mil
%       Phone:  (408) 646-3044/2772
%*******************************************************************
% !!!!!!!! This routine is only for PHASE-10 !!!!!!!!!
% Required M=files:
%              wav2d.m  --  grandparent routine
%              recon2d.m -- parent routine
%              ptzro2d.m -- puts n zeros rows & cols btwn coeff's
%-------------------------------------------------------------------
array=ptzro2d(array,1);
 % generate the workspace
[NHv NHh]=size(mask);
[row col]=size(array);
array=[zeros(row,1) array zeros(row,NHh-2); zeros(NHv-2,col+NHh-1)];
a=0;
 for row1=1:row-1
   a=a+1;
   b=0;
   for col1=1:col
    b=b+1;
    wrk=array(row1:row1+NHv-1,col1:col1+NHh-1);
    x(a,b)=sum(sum(mask.*wrk));
   end
 end
```

```
function x=mmlt01(array,mask)
%*******************************************************************
% MMLT01.M    mask multiplication and summation for the 2-D phase 01
%             reconstruction
%
% By:   LT J. E. Legaspi
%       Professor Lam
%       US Naval Postgraduate School, Monterey CA
%       e-mail: legaspi@ece.nps.navy.mil
%               lam@ece.nps.navy.mil
%       Phone:  (408) 646-3044/2772
%*******************************************************************
% !!!!!!!! This routine is only for PHASE-01 !!!!!!!!!
% Required M=files:
%              wav2d.m  --  grandparent routine
%              recon2d.m -- parent routine
%              ptzro2d.m -- puts n zeros rows & cols btwn coeff's
%-------------------------------------------------------------------
array=ptzro2d(array,1);
 % generate the workspace
[NHv NHh]=size(mask);
[row col]=size(array);
array=[zeros(1,col+NHh-2);arrayzeros(row,NHh-2);zeros(NHv-2,col+NHh-2)];
a=0;
 for row1=1:row
   a=a+1;
   b=0;
   for col1=1:col-1
```

```
    b=b+1;
    wrk=array(row1:row1+NHv-1,col1:col1+NHh-1);
    x(a,b)=sum(sum(mask.*wrk));
   end
  end
```

```
function x=mmlt11(array,mask)
%*********************************************************************
% MMLT11.M    mask multiplication and summation for the 2-D phase 11
%              reconstruction
%
% By:    LT J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:  (408) 646-3044/2772
%*********************************************************************
% !!!!!!!! This routine is only for PHASE-11 !!!!!!!!!
% Required M=files:
%                wav2d.m  --  grandparent routine
%                recon2d.m --  parent routine
%                ptzro2d.m --  puts n zeros rows & cols btwn coeff's
%-------------------------------------------------------------------
array=ptzro2d(array,1);
 % generate the workspace
[NHv NHh]=size(mask);
[row col]=size(array);
array=[zeros(1,col);array];rows=row+1;
array=[zeros(rows,1) array];cols=col+1;
array=[array zeros(rows,NHh-2)];cols=cols+NHh-2;
array=[array; zeros(NHv-2,cols)];
a=0;
 for row1=1:row
   a=a+1;
   b=0;
   for col1=1:col
    b=b+1;
    wrk=array(row1:row1+NHv-1,col1:col1+NHh-1);
    x(a,b)=sum(sum(mask.*wrk));
   end
  end
function x=ptzro2d(input,n)
```

```
%****************************************************************
%PTZRO2D.M   puts "n" rows and columns of zeros after each
%             coefficient in the 2-D input array
%
% By:    LT J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:  (408) 646-3044/2772
%****************************************************************
```

```
% Required routines:
%               wav2d.m    -- great-grandparent routine
%               recon2d.m  -- grand parent routine
%               mmlt??.m   -- parent routine, ??=00,01,10,11
%--------------------------------------------------------------
[r c]=size(input);
x0=[];
for j=1:c
  x0=[x0 input(:,j) zeros(r,n)];
end
[r c]=size(x0);
x=[];
for j=1:r
  x=[x;x0(j,:);zeros(n,c)];
end
```

```
%*****************************************************************
% MP2D.M  2-D Multi-phase decomposition
%
% By:    LT J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:  (408) 646-3044/2772
%*****************************************************************
% MP2D conducts a multiple phase decomposition routine for the 2-D
% case. The following m-files are required:
%
%     wav2d.m      - calling routine
%     mpdcmp2d.m - multiple phase decomp for 1 resolution level
%     strplt.m     - metafile storage of selected plots
%     enrg2dmp.m - multiple phase energy check
%-----------------------------------------------------------------
% Variables are the same as the calling program
%
%-----------------------------------------------------------------
%--------------------------------------------------------------
% Initialization of the routine
clc
clg
subplot(211),mesh(c0),title('3-DPlot of the Image Array')
subplot(212),contour(c0),title('ContourDisplay of the Image')
pause
strplt
clg
clg
axis('normal')
hold off
disp(['     '])
lowest=input('Enter the lowest resolution level:  ');
lowest=-abs(lowest);
m=0;
%--------------------------------------------------------------
%--------------------------------------------------------------
% Calculating the Coefficients for each resolution level
while m>=lowest+1
   eval(['c',num2str(-m+1),'=mpdcmp2d(c',num2str(abs(m)),',HhHv,m);'])
   eval(['d1',num2str(-m+1),'=mpdcmp2d(c',num2str(abs(m)),',HhGv,m);'])
   eval(['d2',num2str(-m+1),'=mpdcmp2d(c',num2str(abs(m)),',GhHv,m);'])
   eval(['d3',num2str(-m+1),'=mpdcmp2d(c',num2str(abs(m)),',GhGv,m);'])
    %  Display of the coefficients
   subplot(221),eval(['mesh(c',num2str(abs(m)+1),')'])
     xlabel(['c'])
     title(['wavelet horiz:  ',waveleth])
   subplot(222),eval(['mesh(d1',num2str(abs(m)+1),')'])
     xlabel(['d1'])
     title(['wavelet vert:  ',waveletv])
   subplot(223),eval(['mesh(d2',num2str(abs(m)+1),')'])
     title(['d2'])
     xlabel('multiphase case')
   subplot(224),eval(['mesh(d3',num2str(abs(m)+1),')'])
     title(['d3'])
     xlabel(['Resolution level ',num2str(m-1)])
    pause
    strplt
     clg
   subplot(221),eval(['contour(c',num2str(abs(m)+1),')'])
      xlabel(['c'])
      title(['wavelet horiz:  ',waveleth])
   subplot(222),eval(['contour(d1',num2str(abs(m)+1),')'])
      xlabel(['d1'])
      title(['wavelet vert:  ',waveletv])
   subplot(223),eval(['contour(d2',num2str(abs(m)+1),')'])
     title(['d2'])
      xlabel('multiphase case')
   subplot(224),eval(['contour(d3',num2str(abs(m)+1),')'])
     title(['d3'])
      xlabel(['Resolution level ',num2str(m-1)])
    pause
    strplt
    clg
  m=m-1;
end
%-------------------------------------------------------------
%-------------------------------------------------------------
%  Energy check for the 2-D multiphase case
enrg2dmp
%-------------------------------------------------------------
%    The End




function x=mpdcmp2d(array,mask,m)
%*****************************************************************
% MPDCMP2D.M  2-D Multi-phase decomposition decomposition function
%             m-file
%
% By:    LT  J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:  (408) 646-3044/2772
%*****************************************************************
% MPDCMP2D is the actual function m-file that zero pads the input
%          image array given the size of the mask operator and the
%          resolution level m. "m" is the current resolution level
%          and is a negative number.
%
%    mp2d.m     -  calling routine
%
%-----------------------------------------------------------------
```

```
[NHv NHh]=size(mask);
[a b]=size(array);
x=zeros(a+2^(-m)*(NHv-1),b+2^(-m)*(NHh-1));
for k=1:NHh
 for l=1:NHv
          %add the rows of zeros
  work=[zeros(2^(-m)*(l-1),b);array;zeros((NHv-l)*2^(-m),b)];
  [a b]=size(work);
          %add the cols of zeros
  work1=[zeros(a,2^(-m)*(k-1))work zeros(a,(NHh-k)*2^(-m))];
  work2=mask(NHv-l+1,NHh-k+1)*work1;
  x=x+work2;
% clear work work1 work2
 end
end
```

```
%*************************************************************
% enrg2dmp.m      Determining the energy in each resolution level
%                 multiphase case
%
% By:    LT J. E. Legaspi
%        Professor Lam
%        US Naval Postgraduate School, Monterey CA
%        e-mail:  legaspi@ece.nps.navy.mil
%                 lam@ece.nps.navy.mil
%        Phone:   (408) 646-3044/2772
%*************************************************************
% In this MULTIPHASE case, the lower level will have
% four possible phases, thus its energy will be 4 times as high as
% the next higher level so we will divide the energy of the lower
% level and the value should equal the energy of the c's of the
% next higher level.
%-------------------------------------------------------------
%  Variables (other than defined in mp2d.m):
%
%    E    - energy array for [c,d1,d2,d3] for eaach level
%    esum -  total energy of each level: [c(m)+d1+d2+d3]/4=c(m+1)
%-------------------------------------------------------------
E=zeros(-lowest+1,4);
% Initialization of the energy for resolution level 0
E(1,1)=sum(sum(c0.^2));
lvl=-1;
k=1;
% Calculate the energies for each resolution level
while lvl>=lowest
  E(k+1,1)=sum(sum((eval(['c',num2str(abs(lvl))])).^2*4^(lvl)));
  E(k+1,2)=sum(sum((eval(['d1',num2str(abs(lvl))])).^2*4^(lvl)));
  E(k+1,3)=sum(sum((eval(['d2',num2str(abs(lvl))])).^2*4^(lvl)));
  E(k+1,4)=sum(sum((eval(['d3',num2str(abs(lvl))])).^2*4^(lvl)));
  lvl=lvl-1;
  k=k+1;
end
E=E/E(1,1);  % normalize the energy to level 0
E=flipud(E); % top row is lowest resolution level
% Display of the energies by coefficients
clg
k=[lowest-1 1 0 1.1];
axis(k);subplot(221),bar(lowest:1:0,E(:,1))
        xlabel('c')
        title(['wavelet horiz:  ',waveleth])
axis(k);subplot(222),bar(lowest:1:0,E(:,2))
       xlabel('d1')
       title(['wavelet vert:  ',waveletv])
axis(k);subplot(223),bar(lowest:1:0,E(:,3))
       title('d2')
       xlabel('multiphase case')
axis(k);subplot(224),bar(lowest:1:0,E(:,4))
        title('d3')
pause
strplt
clg
clc
E=E';
esum=sum(E);
esum=sum(E);
disp(['     ']')
dd=['     Verification of the energy in each resolution level     '
    '                                                              '
    '                                                              '
    'Level    Energy in          Energy in C         Difference'
    '  m      level m-1 =          level m                     '
    '          c+d1+d2+d3                                      '
    '--------------------------------------------------------------'];
disp(dd)
j=lowest;
for i=1:-lowest
  a=esum(i);b=E(1,i+1);d=abs(a-b);c=j+1;
  fprintf('%2.0f     %17.15f     %17.15f     ',c,a,b)
  fprintf('%17.15f\n',d)
  j=j+1;
end
pause
clc
%                    End
```

```
%*******************************************************************
% IDATA.M     Sample Image data for testing the routines
%
% By:   LT J. E. Legaspi
%       Professor Lam
%       US Naval Postgraduate School, Monterey CA
%       e-mail:  legaspi@ece.nps.navy.mil
%                lam@ece.nps.navy.mil
%       Phone:  (408) 646-3044/2772
%*******************************************************************
K=menu('Select image','Box1','Box2','Box3','B/W','2B/W','8B/W',...
      'X','Hexagon');


if K==1
% first image is a box
im=zeros(100,100);
im(20:21,20:80)=ones(2,61);
im(79:80,20:80)=ones(2,61);
im(22:78,20:21)=ones(57,2);
im(22:78,79:80)=ones(57,2);

elseif K==2
im=zeros(100,100);
im(20,20:79)=ones(1,60);
im(80,20:79)=ones(1,60);
im(20:79,20)=ones(60,1);
im(20:79,80)=ones(60,1);

elseif K==3
im=zeros(100,100);
im(1:2,1:61)=ones(2,61);
im(60:61,1:61)=ones(2,61);
im(3:59,1:2)=ones(57,2);
im(3:59,60:61)=ones(57,2);

elseif K==4
% second image is a 1 white and 1 black rectangle
im=zeros(100,100);
im(:,51:100)=ones(100,50);

elseif K==5
% third image are two black boxes and two white boxes
im=zeros(100,100);
im(1:50,1:50)=ones(50,50);
im(51:100,51:100)=ones(50,50);

elseif K==6
% fourth image are 8 black and 8 white boxes
im=zeros(100,100);
im(1:25,:)=[ones(25,25)zeros(25,25) ones(25,25) zeros(25,25)];
im(26:50,:)=[zeros(25,25)ones(25,25) zeros(25,25) ones(25,25)];
im(51:75,:)=[ones(25,25)zeros(25,25) ones(25,25) zeros(25,25)];
im(76:100,:)=[zeros(25,25)ones(25,25) zeros(25,25) ones(25,25)];

elseif K==7
% fifth image are 2 45 degree diagonals
im=eye(100,100);
im=sign(im+flipud(im));

elseif K==8
% sixth image is a hexagon
for row=1:20
  n=2*row-1;
  im6(row,n:n+2)=ones(1,3);
end
[a b]=size(im6);
row2=ceil(sqrt(a^2+b^2));
im6a=zeros(row2,b);
im6a(:,b-1:b)=ones(row2,2);
rim=[im6;im6a;fliplr(im6)];
lim=fliplr(rim);
im6=[lim rim];
clear row row2 im6a lim rim
[a b]=size(im6);
im6a=zeros(100,100);
aa=floor((100-a)/2);
bb=floor((100-b)/2);
im6a(aa+1:aa+a,bb+1:bb+b)=im6;
im=im6a;
clear im6 im6a a aa b bb K n
end
```

C. ROUTINES COMMON TO BOTH DIMENSIONS

```
%*****************************************************************
% chkputer.m    checks for the proper operating system
%
% By:   LT J. E. Legaspi
%       Professor Lam
%       US Naval Postgraduate School, Monterey CA
%       e-mail:  legaspi@ece.nps.navy.mil
%                lam@ece.nps.navy.mil
%       Phone:  (408) 646-3044/2772
%*****************************************************************
%
% This routine reads the variable "cmptr" to see if it can run in
% the current system.  At this time, the algorithm will run on a
% PC386, UNIX, or SUN system.  If the operating system is a VAX-
% based, the routine will not be as reliable since the VAX MATLAB
% does not recognize IEEE arithmetic.  The routine also deletes all
% other leg*.met files prior to running.
%-----------------------------------------------------------------

[cmptr mxsz]=computer;
a=length(cmptr);
clc;

  if     cmptr(1)=='P'& cmptr(2)=='C'&cmptr(3)=='3',
         !del leg*.met;
  elseif cmptr(1)=='S'&cmptr(2)=='U'&cmptr(3)=='N',
         !rm leg*.met;
  elseif cmptr(a-3)=='U'&cmptr(a)=='X',
         !rm leg*.met;
  else
         clc
         disp(['      '])
         q1=['*****Error in input******'
             'Hardware cannot support  '
             'the algorithm...PLS      '
             ' Restart the Program     '];
         disp(q1)
         return
  end




%*****************************************************************
% strplt.m
%
% Store Plot M-file for wavelet decomposition routine
%
% By:   LT J. E. Legaspi
%       Professor Lam
%       US Naval Postgraduate School, Monterey CA
%       e-mail:  legaspi@ece.nps.navy.mil
%                lam@ece.nps.navy.mil
%       Phone:  (408) 646-3044/2772
%*****************************************************************
%
% This program stores the plot as a metafile with the prefix "leg"
% followed by a number starting from 0 on upwards and ending with
% the extension ".met" You must be familiar with GPP to get hard-
% copies of the plots. "pltcnt" is the only variable necessary for
% the proper indexing of the leg*.met files.  Ensure that the
% calling program does not redefine the variable!
%------------------------------------------------------------------

disp([' ']')
q5=input('Store the Plot (Y/N)? ','s');
 if q5 == 'Y' | q5 == 'y',
 eval(['meta leg',num2str(pltcnt)])
 pltcnt=pltcnt+1;
 end



%**************************************************************
%       daubdata.m
%
% By:   CAPT/USMC  Kalmbach, M.
%       Professor Lam
%       US Naval Postgraduate School, Monterey CA
%       e-mail:  lam@ece.nps.navy.mil
%       Phone:   (408) 646-3044
%
%
function[hcoeff] = daubdata(x)
% This function returns the proper "h" coefficients for the specified order
% Daubechies wavelet. The input value of "x" is the specified order.
if x == 2
 hcoeff = [0.482962913145;0.836516303738;0.224143868042;-0.129409522551];
elseif x == 3
 hcoeff = [0.332670552950;0.806891509311;0.459877502118;-0.135011020010;
          -0.085441273882;0.035226291882];
elseif x == 4
 hcoeff = [0.230377813309;0.714846570553;0.630880767930;-0.027983769417;
          -0.187034811719;0.030841381836;0.032883011667;-0.010597401785];
elseif x == 5
 hcoeff = [0.160102397974;0.603829269797;0.724308528438;0.138428145901;
          -0.242294887066;-0.032244869585;0.077571493840;-0.006241490213;
          -0.012580751999;0.003335725285];
elseif x == 6
 hcoeff = [0.111540743350;0.494623890398;0.751133908021;0.315250351709;
          -0.226264693965;-0.129766867567;0.097501605587;0.027522865530;
          -0.031582039318;0.000553842201;0.004777257511;-0.001077301085];
elseif x == 7
 hcoeff = [0.077852054085;0.396539319482;0.729132090846;0.469782287405;
          -0.143906003929;-0.224036184994;0.071309219267;0.080612609151;
          -0.038029936935;-0.016574541631;0.012550998556;0.000429577973;
          -0.001801640704;0.000353713800];
elseif x == 8
 hcoeff = [0.054415842243;0.312871590914;0.675630736297;0.585354683654;
          -0.015829105256;-0.284015542962;0.000472484574;0.128747426620;
          -0.017369301002;-0.044088253931;0.013981027917;0.008746094047;
          -0.004870352993;-0.000391740373;0.000675449406;-0.000117476784];
elseif x == 9
 hcoeff = [0.038077947364;0.243834674613;0.604823123690;0.657288078051;
          0.133197385825;-0.293273783279;-0.096840783223;0.148540749338;
          0.030725681479;-0.067632829061;0.000250947115;0.022361662124;
          -0.004723204758;-0.004281503682;0.001847646883;0.000230385764;
          -0.000251963189;0.000039347320];
elseif x == 10
 hcoeff = [0.026670057901;0.188176800078;0.527201188932;0.688459039454;
          0.281172343661;-0.249846424327;-0.195946274377;0.127369340336;
          0.093057364604;-0.071394147166;-0.029457536822;0.033212674059;
          0.003606553567;-0.010733175483;0.001395351747;0.001992405295;
          -0.000685856695;-0.000116466855;0.000093588670;-0.000013264203];
```

```
else
  hcoeff = [];
  break
end
hcoeff = hcoeff'/sqrt(2);              % normalize the "h" vector
return
```

```
*************************************************************************
#! /bin/csh -f
#
#************************************************************************
# meta14        C-Shell Routine
#
# meta14 is a command executable file for the generation of
# the leg*.met metafiles into postscript graphic files for the
# SPARC printer number 14 on the fourth floor of Spanagel 431.
# This routine is system dependent and is guaranteed to work
# on the UNIX based system of the ECE Department at the Naval
# Postgraduate School.
#************************************************************************
#
ls -1 leg*.met >tmmmp1
echo "#! /bin/csh -f " > tmmmp2
awk -F. '{printf "gpp " $1 " -dps -ol \n" ;\
        printf "lpr14 " $1 ".ps \n";\
        printf "rm " $1 ".ps \n"}' tmmmp1 > >tmmmp2
chmod 0700 tmmmp2
tmmmp2
rm tmmmp1 tmmmp2
```

LIST OF REFERENCES

1. Mallat, S.G., "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 11, n. 7, pp. 674-693, July 1989.

2. Daubechies, I., "Orthonormal Bases of Compactly Supported Wavelets," *Communications on Pure and Applied Mathematics*, v. XLI, pp. 909-996, 1988.

3. Lam, A.W. and Legaspi, J.L., "Multiple-Phase Discrete Wavelet Transform," Department of Electrical and Computer Engineering, Naval Postgraduate School, Monterey, California, July 1992, submitted for publication.

4. Kalmbach, M., "Wavelet-based Multiresolution Analyses of Signals," Master's Thesis, Naval Postgraduate School, Monterey, California, June 1992.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
4.	Professor Alex W. Lam, Code EC/La Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	5
5.	Professor Herschel H. Loomis, Code EC/Lm Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
6.	Commander Naval Space Command ATTN: Code N155 Dahlgren, VA 22448-5170	1
7.	Naval Research Lab ATTN: Code 9120, CAPT Mitschang 4555 Overlook Ave, SW Washington, DC 20375	1
8.	LT Legaspi, Joey E. Patrol Squadron Special Projects Unit Two Naval Air Station Barbers Point HI 96862-6100	2